UNIVERSITATEA TEHNICA GH. ASACHI IASI
FACULTATEA DE MECANICA
CATEDRA ORGANE DE MASINI SI MECATRONICA
Disciplina: Actionarea sistemelor mecatronice

LUCRARE DE LABORATOR NR. 1

MODULAR PRODUCTION SYSTEM (MPS)– ECHIPAMENT DE INSTRUIRE IN MECATRONICA
Prezentare STATIE DE DISTRIBUTIE

1. Scopul lucrarii:

- cunoasterea hardware a unui sistem mecatronic cu actionare pneumatica (hidronica);
- cunoasterea tehnicii PLC (Programmable logic Controlers);
- prezentarea competentelor ce pot fi dezvoltate pe un astfel de sistem;

2. Introducere

Stațiile MPS sunt elemente centrale ale sistemelor modulare de producție. Fiecare stație are unul sau mai multe module legate între ele printr-o plăcuță universală de aluminiu cu profil plat şi prin controlere PLC. Ele ajută la atingerea pregătirii profesionale într-un mod clar şi eficient, deoarece oferă posibilitatea de studiu într-un cadru cât mai real, studiul efectuându-se pe componente cât mai reale.

Avantaje:

- Stațiile pot fi combinate şi potrivite individual;
- Interfața este standard şi unică;
- Sistemul deschis noilor tehnologii - comunicații pe bază de bus şi vizualizare a procesului;
- Sistemul deschis noilor idei - definirea interfețelor este, de exemplu, baza noilor proiecte de lucru;
- Sistemul deschis noilor cerințe de calificare - munca în echipă.

Sistemele MPS sunt legate prin conectori electrici şi mecanici, prin interfețe bine definite. În funcție de combinația stațiilor din standul din fig.1. se pot produce piese cilindrice funcționale formate din:

- Corp cilindric
- Piston
- Arc
- Capac



În componența standului intră următoarele stații MPS:

- Stația de distribuție;
- Stația de testare;
- Stația de procesare;
- Stația de manevrare-asamblare;
- Stația de memorare;
- Ansamblu stație cu robot;
- Stație de testare a funcțiilor;
- Stație de sortare;
- Stație de presare hidraulică.

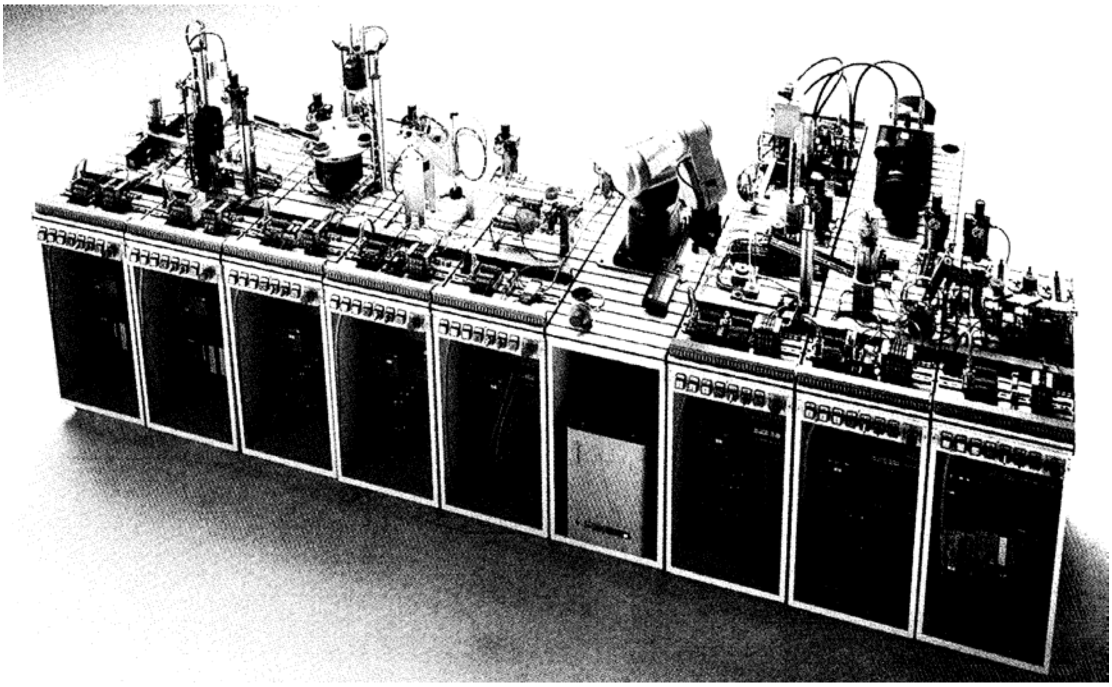


Fig. 1.

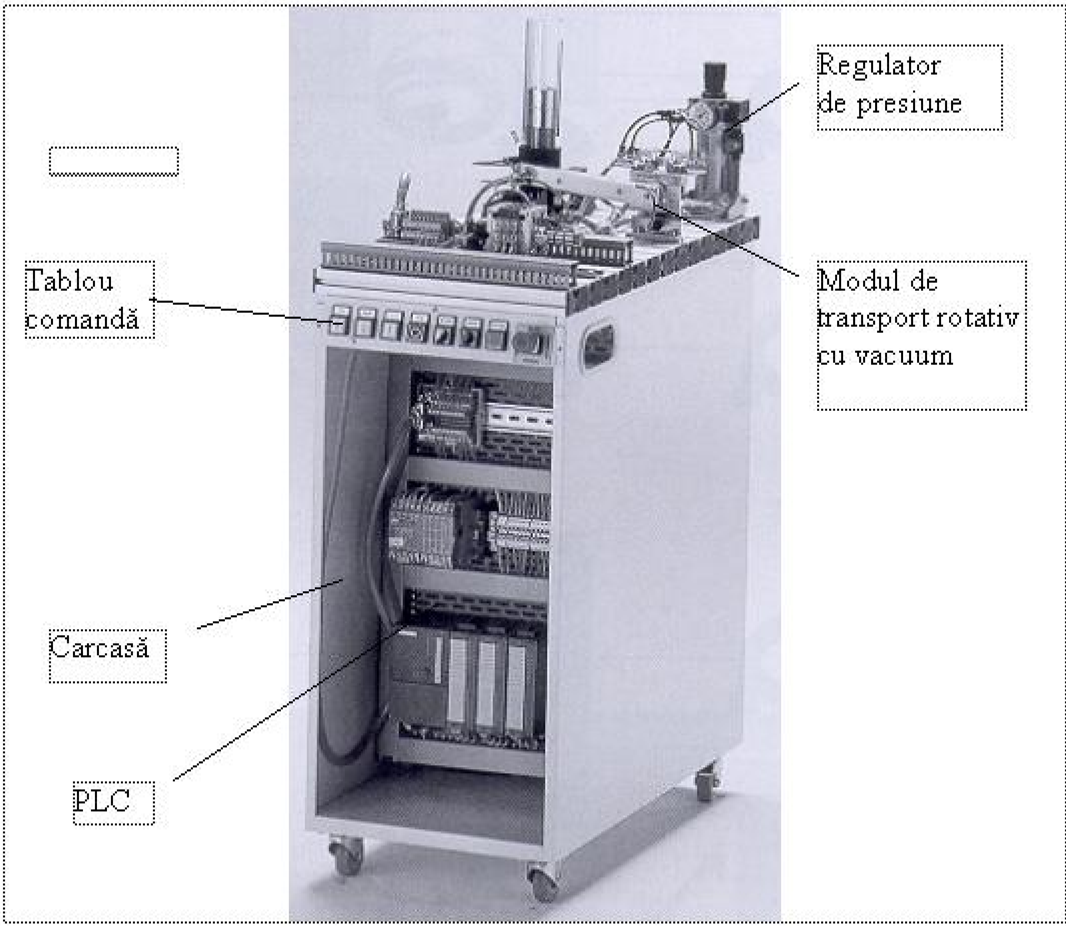
Ordinea stațiilor corespunde procesului tehnologic. Corpurile cilindrilor sunt distribuite, testate şi plasate în continuare cu ajutorul dispozitivelor de testare.

Robotul asamblează cilindrul pneumatic folosind corpul, pistonul, arcul şi capacul care sunt făcute la o presa hidraulică. Cilindrii finisați sunt testați din punct de vedere al funcționalității şi sortați după: tip, culoare, mărime.

In dotarea laboratorului de *Acționarea sistemelor mecatronice* există un modul MPS, denumit **Stație de distribuție** (Distribution station) (fig. 1).

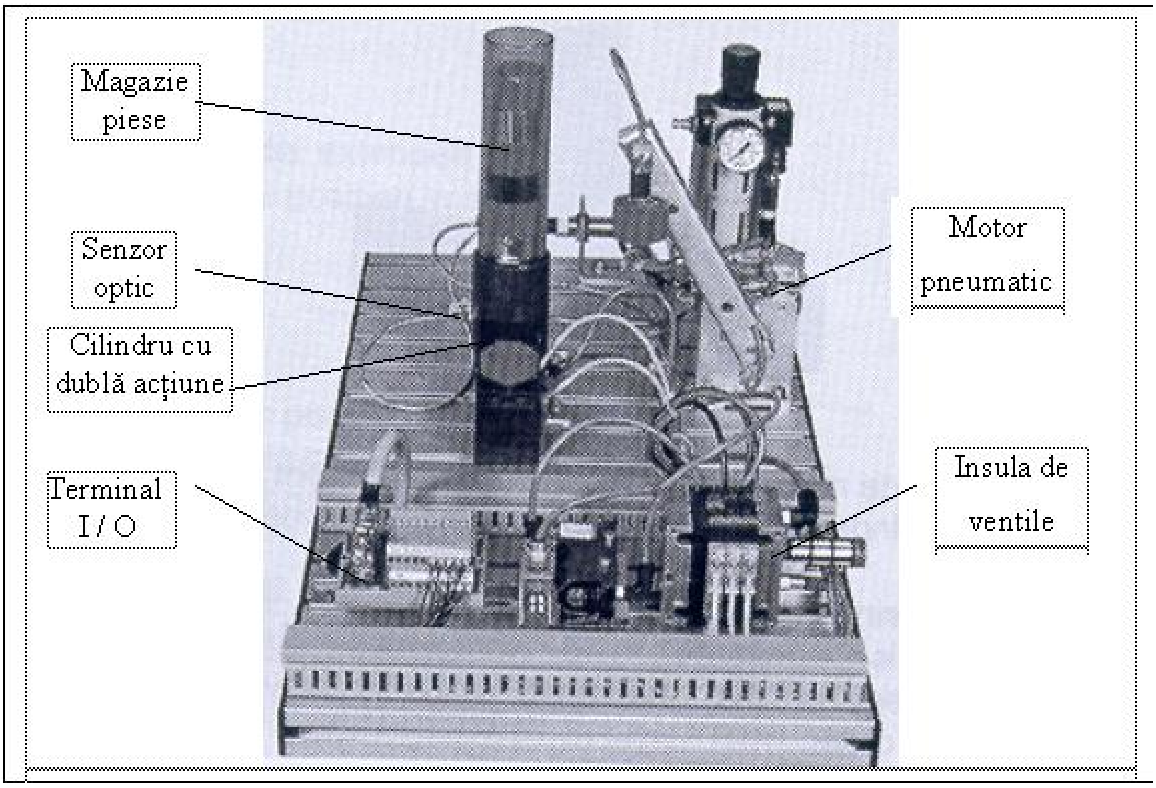
3. Elementele componente ale (MPS)-Statie de distributie

Componentele Stației de distribuție şi competențele dezvoltate prin instruire sunt prezentate în fig. 2., fig. 3. şi tabelul 1.



Regulator de presiune
Tablou comandă
Modul de transport rotativ cu vacuum
Carcasă
PLC

Fig. 2



Magazie piese
Senzor optic
Cilindru cu dublă acțiune
Terminal I / O
Motor pneumatic
Insula de ventile

Fig. 3

Competența dezvoltată	Scopul instruirii
Sisteme mecanice	- dobândirea de cunoştințe despre componentele unui sistem mecanic; - asamblarea si dezasamblarea componentelor mecanice; - proiectarea şi aşezarea componentelor mecanice într-un sistem.
Pneumatică	- dobândirea de cunoştințe şi utilizarea componentelor pneumatice; - citirea si construirea unor circuite pneumatice; - dobândirea de cunoştințe şi utilizarea tehnologiei cu vacuum; - optimizarea si reglarea circuitelor pneumatice (viteza, siguranța etc.); - realizarea practică a unor circuite pneumatice.
Inginerie Electrică/Electronică	- instalarea şi testarea componentelor electrice; - citirea şi proiectarea diagramelor circuitelor electrice; - dobândirea de cunoştințe despre circuitele electronice şi utilizarea lor; - dobândirea de cunoştințe despre interfețe şi utilizarea lor.
Senzori	- dobândirea de cunoştințe despre limitatori şi senzori şi utilizarea lor; - dobândirea de cunoştințe despre diferitele tipuri de senzori (optic etc.).
Tehnologia PLC	- dobândirea de cunoştințe despre structura si modul e operare a PLC; - programarea PLC;

	- utilizarea PLC împreună cu un sistem de vizualizare.
Automatizare	- dobândirea de cunoştinţe despre sistemele automate si structura lor; - proiectarea unor sisteme automate; - Combinarea şi optimizarea diferitelor tehnologii pentru un scop specific.
Metrologie	- măsurarea variabilelor electrice şi neelectrice.
Tehnologia manipulatoarelor	- dobândirea de cunoştinţe despre manipulatoare; - programarea manipulatoarelor utilizând PLC.
Reglaje	- adaptarea componentelor electrice si mecanice; - adaptarea senzorilor şi limitatorilor în sistemele automate.
Întreținere, optimizare si depanare	- optimizarea unui sistem automat (durata unui ciclu, siguranță); - localizarea şi repararea defectelor prin metode sistematice.
Competente metodologice	- utilizarea metodelor creative şi sistemelor de planificare; - organizarea şi planificarea lucrului cu paşi.
Abilități sociale	- lucrul practic în echipă; - comunicarea între echipe;

4. Concluzii

- Stația de distribuție este ideală pentru instruirea în sistemele moleculare de producție;

- Atenția este atrasă de aplicarea diferitelor valve de control, cilindri pneumatici şi limitatorilor de cursă a cilindrilor, precum şi tehnologia de vacuum şi valvele de presiune;

Obiectivele atinse:

- Separarea unei piese dintr-o magazie;
- Alimentarea sistemului cu piese separate pentru obținerea unui proces tehnologic.

UNIVERSITATEA TEHNICA GH. ASACHI IASI
FACULTATEA DE MECANICA
CATEDRA ORGANE DE MASINI SI MECATRONICA
Disciplina: Actionarea sistemelor mecatronice

LUCRARE DE LABORATOR NR. 2

MODULAR PRODUCTION SYSTEM (MPS)– ECHIPAMENT DE INSTRUIRE IN MECATRONICA
Mediul de programare FSTFEC pentru automatul programabil FEC-20-AC

1. Scopul lucrarii:

- Cunoastrea mediului de programare FSTFEC care permite elaborarea de programe pentru automatul programabil FEC-20-AC

2. Introducere

Numele de FST FEC vine de la abrevierile denumirilor *Festo Software Tools* şi *Front End Controler*. FST este un mediu de programare care a fost dezvoltat de FESTO pentru mai multe tipuri de automate programabile şi calculatoare industriale pe care firma le produce. Principala caracteristică a acestui mediu este prezența unui sistem de *meniuri standard* pentru toate variantele în care este disponibil. Variantele de software FST depind de aparatul căruia îi este destinat, acestea având ca particularități seturile de comenzi pentru configurare şi limbajele de programare utilizate. Pentru FEC sunt disponibile limbajele de programare *Statement List* (STL) şi *Ladder Diagram* (LDR). Despre limbajele de programare se va discuta în capitolul 6.

Mediul de programare FSTFEC trebuie instalat pe un PC obişnuit. Pentru o instalare cu succes, calculatorul trebuie să dețină un minim de caracteristici. Aceste caracteristici sunt enumerate mai jos:
- hard disk şi floppy disk de 3" (1.44 MB)
- o memorie de cel puțin 512 KB; pentru proiecte mari se recomandă minim 640KB;
- sistem de operare MS-DOS, Windows'95 sau mai nou;
- o interfață serială pentru conectarea cu automatul programabil;
- o interfață paralelă pentru imprimantă (de preferat);
- o altă interfață serială pentru mouse (de preferat);

Programul se poate instala în directorul rădăcină sau în orice alt director. După instalare programul poate fi rulat din directorul în care se află fişierul 'FSTFEC.EXE', cu comanda 'FSTEC', urmată de apăsarea tastei Enter. Următoarele directoare sunt create la instalarea mediului FSTFEC:
- RUNTIME.FEC – conține fişiere cu sistemul de operare la automatului programabil;
- FBLIB.FEC – conține fişiere cu module de funcții predefinite de producător;
- BAUSTEIN.MSC – conține fişiere cu librării şi exemple pentru programarea în C;
- DOC – conține documentație pentru utilizarea FEC-ului.

În afara acestor directoare, FSTFEC mai are nevoie de un *director de proiecte* în care vor fi păstrate programele create de utilizator. La prima rulare a programului FSTFEC se va cere utilizatorului să introducă calea şi numele directorului de proiecte.

3. MEDIUL DE PROGRAMARE FSTFEC

3.1. Descrierea meniurilor.

Imediat după comanda 'FSTFEC' apare o fereastră 'logo' în care este scris cu inițiale mari: FST. Apăsarea oricărei taste duce la încărcarea meniului principal al mediului de programare. Meniul principal este prezentat în figura 1.
Pentru lucrul cu meniurile se poate utiliza:

- mouse-ul;
- tastele speciale F1, F2,…, F8 şi tastele direcţionale: ←, ↑, ↓, →.

Acţiunile declanşate de tastele F1 – F8 sunt descrise la fiecare moment în bara de stare de la baza ecranului (josul ferestrei). Ca regulă, terminarea unei acţiuni şi ieşirea dintr-un submeniu sau din programul principal se face prin apăsarea tastei F8.

Submeniul *Project Management*, fig. 2, cuprinde comenzi pentru: selectarea, crearea, ştergerea, printarea unui proiect, încărcarea acestuia în automatul programabil, salvarea pe harddisk-ul calculatorului a unui proiect aflat în automatul programabil şi inserarea în proiectul curent a unor fişiere.

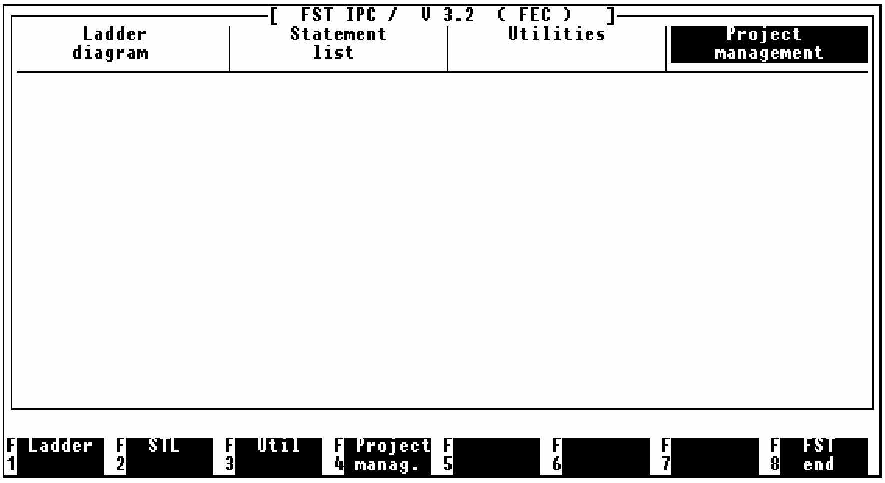


Fig. 1. *Meniul principal al mediului de programare FSTFEC.*

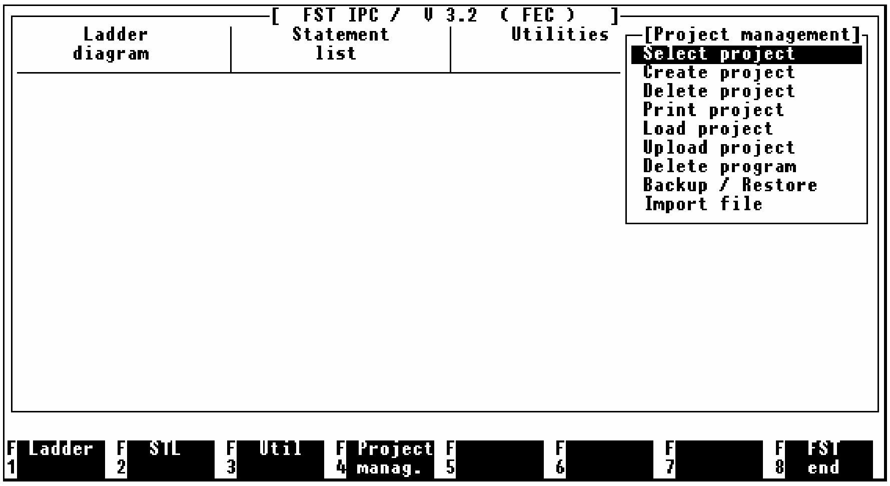


Fig. 2. *Submeniul Project Manegement al mediului de programare FSTFEC.*

Submeniul *Utilities*, fig. 3, conţine comenzi şi acţiuni necesare în pentru:
- alcătuirea listei de alocare a variabilelor absolute (paragraful 5.6);
- editare de texte pentru documentaţia proiectului;
- editarea titlului şi a antetului fiecărei pagini pentru textul scris în mediul FSTFEC;
- configurarea calculatorului şi a automatului programabil;
- configurarea intrărilor şi ieşirilor FEC-ului;
- configurarea driver-elor de comunicaţie pentru cazul când FEC-ul se utilizează în reţea;
- conectarea calculatorului cu automatul programabil pentru acţiuni de depanare on-line;
- printare a documentelor ataşate proiectului selectat.

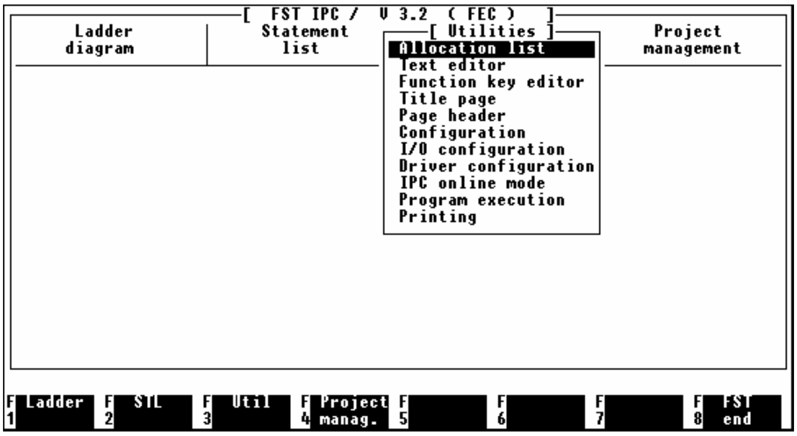


***Fig. 3**. Submeniul Utilities al mediului de programare FSTFEC.*

Submeniul *Statement List*, fig. 4, conține comenzi şi acțiuni necesare creării, modificării şi depanării programelor STL din cadrul proiectului curent.

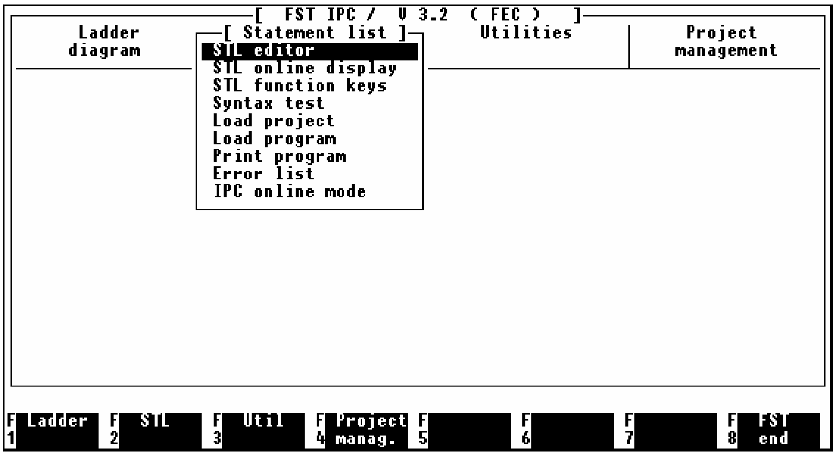


***Fig. 4**. Submeniul Statement List al mediului de programare FSTFEC.*

3.2. Crearea unui proiect.

În mediul de programare FST, un *proiect* este o unitate de control al programelor. Proiectul poate să conțină unul sau mai multe programe şi blocuri de funcții. Pe lângă programe, un proiect poate să conțină o listă de alocare şi un text cu documentația proiectului. Pentru a se putea lucra în mediul FST este neapărat necesar să se creeze un proiect. Este indicat să se creeze un proiect nou pentru fiecare nouă aplicație de automatizare.

Pentru crearea unui proiect se selectează opțiunea *Create project* din meniul *Project Management*. În fereastra care apare, fig. 5, se introduce, de către utilizator, numele proiectului şi, opțional, un comentariu pentru descrierea aplicației. La sfârşit se apasă tasta F1 pentru confirmarea comenzii.

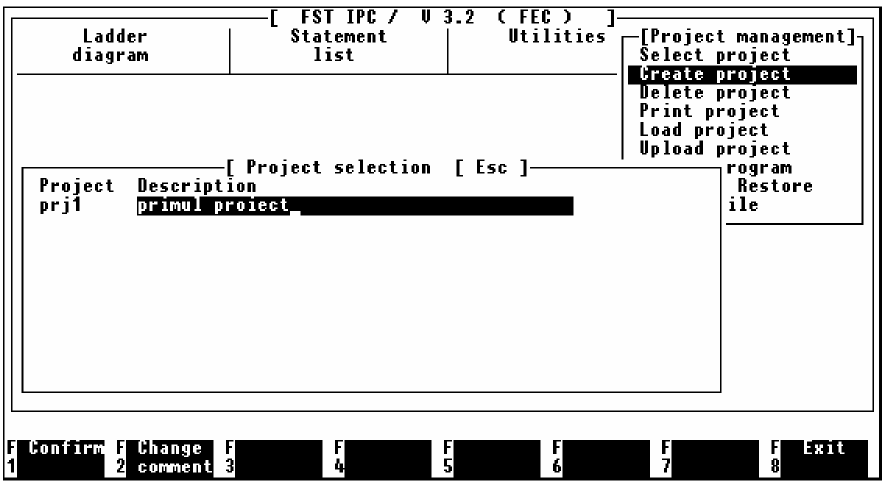


***Fig. 5**. Fereastra de creare a unui proiect.*

3.3. Configurarea mediului de programare FST.

Mediul de programare FST trebuie să facă posibilă comunicarea între ele a trei echipamente hardware: calculatorul, automatul programabil si imprimanta. Setările care se fac în ferestrele deschise de opțiunea *Configuration* din submeniul *Utilities*, asigură o comunicare corectă între aceste echipamente. În timpul configurării, dacă trebuiesc introduse valori numerice, se poate apăsa tasta F9 pentru a obține informații în legăură cu valorile admise în acel câmp de date. Configurarea se poate face pentru:

- calculator;
- comunicarea cu automatul programabil;
- modul de comportare în funcționare al automatului programabil;
- selectarea imprimantei si a caracterelor de tipărire la imprimantă;
- modul de rulare a programelor.

Configurarea calculatorului cuprinde:

- selectarea directorului pentru salvarea proiectelor. Dacă nu se specifică un director anume directorul implicit (creat de program) este 'C:\FESTO';
- selectarea de programe care să se execute imediat înainte de intrarea în mediul FST;
- selectarea de programe care să se execute imediat după iesirea din mediul FST;
- selectarea controler-ului video pentru monitor dintr-un set recunoscut de mediul FST (această setare este de interes numai pentru calculatoare vechi sau necompatibile IBM);
- setarea mouse-ului.

Observație. Dacă s-a executat o setare gresită în configurarea calculatorului (în special pentru placa video) si monitorul apare negru, trebuie repornit calculatorul si, din mediul DOS, sters fisierul 'KONFIG.IPC' care se află în directorul în care a fost instalat mediul FST. După această operație se reintră în mediul FST si se reface configurarea.

Configurarea comunicării cu automatul programabil are în vedere stabilirea corectă a parametrilor de comunicație între calculator si FEC. Se poate alege între porturile seriale COM1 si COM2. La portul selectat se va monta cablul de legătură. Odată ales portul serial se setează viteza de transmisie dintr-un set de valori permise de automatul programabil. Viteza maximă de transmisie, care este si cea implicită, este de 9600 baud (1 baud reprezintă o viteză de transmisie de 1 bit/secundă).

Configurarea modului de comportare în funcționare al automatului programabil se poate face apăsând tasta F6 (IPC mode) după ce s-a selectat opțiunea *Configuration* din meniul *Utilities*.

Pentru opțiunea *Machine function? Y/N*:

- dacă se selectează *Y (Yes)* AP-ul se va comporta ca o *masină*. La comporatarea ca o masină, atunci când FEC-ul este oprit (datorită comutatorului START/STOP sau al unei erori) toate iesirile sunt setate la 0 logic. Când FEC-ul este repornit, se vor rula programele de la început. Cu alte cuvinte, nu se păstrează starea din momentul opririi.
- dacă se selectează *N (No)* AP-ul se va comporta ca o *unitate*. La repornire programele se reiau din punctul de oprire, deci starea din momentul opririi se păstrează. Acest modul nu este disponibil pentru FEC.

La opțiunea *Program control* se poate alege o intrare a FEC-ului care să funcționeze ca un comutator START/STOP. Pentru aceasta se introduce adresa absolută a intrării alese de utilizator pentru acest scop.

Opțiunea *Error program number* oferă posibilitatea specificării numărului unui 'program de tratare a erorilor'. Programele sunt notate cu litera 'P' după care urmează un număr (ex. P0, P1, P2, etc.). În cazul apariției unei erori în funcționarea FEC-ului acest program se va executa ciclic atâta timp cât eroarea nu va fi anulată. Programul P0, care este programul

principal, nu poate fi considerat program de tratare a erorilor. De aceea, în cazul în care nu se doreste un asemenea program se va selecta valoarea 0.
Opțiunea *Error output? Y/N* se foloseste pentru specificarea unei iesiri digitale care va fi setată în 1 logic atunci când apare o eroare. Pentru aceasta se selectează *Y (Yes)*. Adresa absolută a iesirii trebuie furnizată în câmpul *Error output number*. Dacă nu se doreste o astfel de indicație a erorilor atunci se selectează *N (No)*.

3.4. Configurarea intrărilor si iesirilor automatului programabil.

Configurarea intrărilor si iesirilor realizează o corespondență între intrările/iesirile *fizice* (externe) ale automatului programabil si cele *numerice* (interne). Dacă această configurare nu se execută (pentru fiecare proiect) atunci nu se pot accesa intrările si iesirile fizice.
Orice semnal de intrare în automatul programabil FEC reprezintă un *bit*. O intrare se notează cu litera 'I' urmată de numărul unui cuvânt si de numărul bitului din componența acestui cuvânt. Cele două numere sunt separate prin punct. De exemplu: intrarea I5.2 înseamnă al trei-lea bit din cuvântul numărul 5 (numărătoarea celor 16 biți ai unui cuvânt se face de la 0 la 15). Dacă se face referire la tot cuvântul numărul 5, atunci se foloseste notația IW5 (I = Input, W = Word). Acelasi lucru este valabil pentru iesiri care se notează cu litera 'O' (Output).
Desi un cuvânt are 16 biți, intrările în FEC (care sunt în număr de 12) au fost grupate de constructor în două cuvinte separate. Primul din aceste cuvinte va conține primele 8 intrări în primii 8 biți. Cel de-al doilea cuvânt va conține ultimele 4 intrări în primii 4 biți. Pentru iesiri se foloseste un singur cuvânt în care primii 8 biți desemnează cele 8 iesiri.
La configurarea intrărilor si iesirilor se stabilesc <u>numerele cuvintelor</u> (word) pe care utilizatorul vrea să le folosească pentru desemnarea adreselor absolute ale intrărilor si iesirilor FEC-ului. Dacă se alege pentru intrare, ca adresă a primului cuvânt, numărul 0 (cel de-al doilea cuvânt va avea automat numărul 1) atunci, în programare, cele 12 intrări vor putea fi accesate cu notațiile:

IW0								IW1			
I0.0	I0.1	I0.2	I0.3	I0.4	**I0.5**	I0.6	I0.7	I1.0	I1.1	I1.2	I1.3

Dacă se alege pentru iesire, de exemplu, ca adresă a cuvântului, numărul 3, în programare cele 8 iesiri se vor putea accesa cu notațiile:

OW3							
O3.0	O3.1	O3.2	O3.3	O3.4	O3.5	O3.6	O3.7

Pentru configurare se alege opțiunea *I/O Configuration* din submeniul *Utilities*. În fereastra care apare se apasă F1 pentru inserarea unui modul (card) de intrări/iesiri. Obligatoriu se selecteaza opțiunea 'FEC'(vezi fig. 6). Adițional se poate selecta opțiunea 'trimmer' pentru cazul folosirii potențiometrului de pe partea frontală a carcasei FEC-ului. La sfârsitul operației de configurare se apasă F1 pentru confirmare, după care se apasă F8 si se selectează opțiunea *Save and quit editor*.

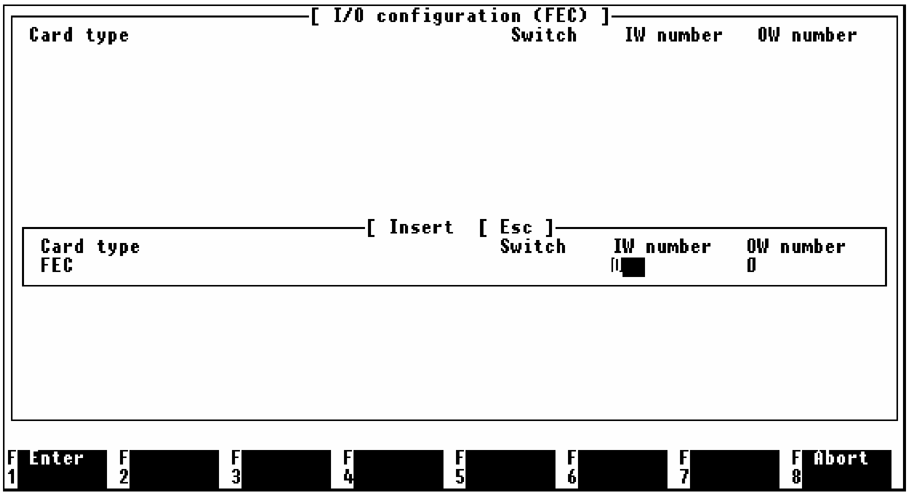


Fig. 6. *Fereastra de configurare a intrarilor/iesirilor pentru FEC.*

4. Crearea unei liste de alocare.

Variabilele de programare (operanzi) desemnează locații de memorie, intrări, iesiri, temporizatoare, numărătoare, etc. Toate aceste date au o adresă internă de forma prezentată la paragraful 5.5. Pentru memorie se foloseste litera F (flag), pentru temporizatoare litera T (timer) iar pentru numărătoare litera C (counter). Literele denotă tipul datei iar numerele constituie adresa din memorie. Exemple:

- FW6 – este cuvântul numărul 6 din memoria automatului programabil;
- F2.1 – este bitul 1 al cuvântului 2 din memorie;
- T1 – este temporizatorul 1 din memorie;
- C4 – este numărătorul 4.

Variabilele din exemplele de mai sus se numesc *operanzi absoluți*.

Mediul de programare FST permite asocierea de *nume simbolice* la datele din memorie. Numele simbolice sunt siruri de caractere care se folosesc în timpul programării si țin locul operanzilor absoluți (notațiilor convenționale ale variabilelor). În acest fel programarea aplicațiilor complexe se poate face mai usor, numele simbolice fiind asociate cu echipamentele şi acțiuni familiare utilizatorului. De exemplu, se pot utiliza asocieri de genul: I0.3 = 'START', O0.2 = 'bec_rosu', T1 = 'pauza_1s', iar în programe, în loc de *I0.3* se va scrie *START*, în loc de *O0.2* se va scrie *bec_rosu*, etc.

Numele simbolice se numesc *operanzi simbolici* si pot fi formate din maxim 9 caractere alfanumerice la care se adaugă caracterul '_' (underscore). Numele simbolice nu trebuie să înceapă cu o cifră si nu trebuie să conțină spații libere.

În figura 7. se poate vizualiza fereastra de introducere a asocierilor în lista de alocare. Această fereastră se deschide după selectarea opțiunii *Allocation List* din submeniul *Utilities*. După deschidere, în bara de stare de la baza ferestrei sunt indicate acțiunile posibile prin apăsarea tastelor F1 – F8.

Principalul avantaj al utilizării listei de alocare consta în scrierea si înțelegerea mai rapidă a programelor. Un al doilea avantaj se referă la necesitatea reactualizării unui program în cazul în care s-au efectuat mici modificări în ordinea semnalelor de intrare sau de iesire. Dacă se folosesc nume simbolice, programul sursă poate să rămâna neschimbat, modificările făcându-se numai în lista de alocare.

Este recomandat ca lista de alocare să se întocmească înainte de scrierea programelor. Oricum, în timpul programării se pot adăuga asocieri noi în lista de alocare. La sfârsitul editării listei de alocare se termină acțiunea apăsând tasta F8 si selectând opțiunea *Save and quit editor*.

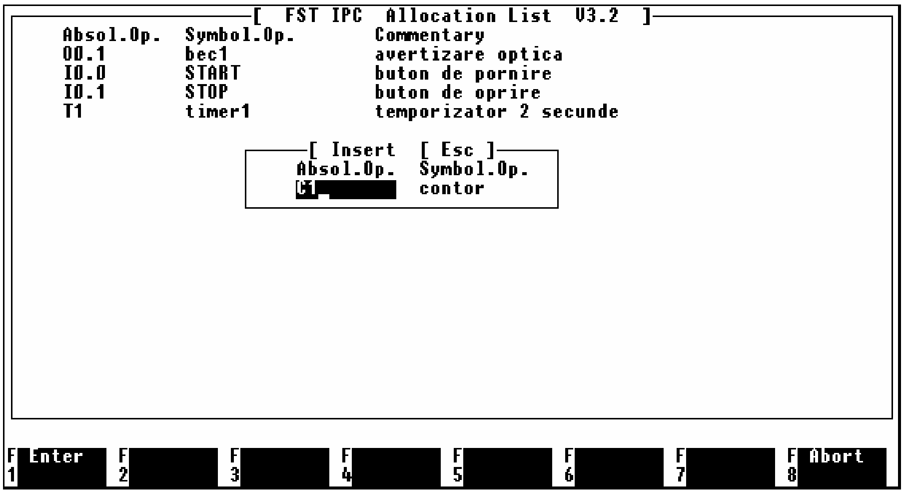


Fig. 7. *Ferestre de editare a listei de alocare.*

5. Crearea unui program.

Pentru editarea de programe STL (sau LDR) se foleseşte submeniul *Statement list (Ladder Diagram)* din meniul principal, fig. 4. Acest submeniu oferă comenzi pentru:

- crearea, scrierea şi modificarea de programe STL (LDR);
- verificarea corectitudinii sintactice a programelor scrise;
- vizualizarea listei de erori în cazul programelor scrise incorect;
- încărcarea unui proiect în automatul programabil;
- printarea separată a programelor;
- urmărirea on-line a execuției unui program;
- conectarea on-line cu automatul programabil pentru vizualizarea datelor din memoria acestuia.

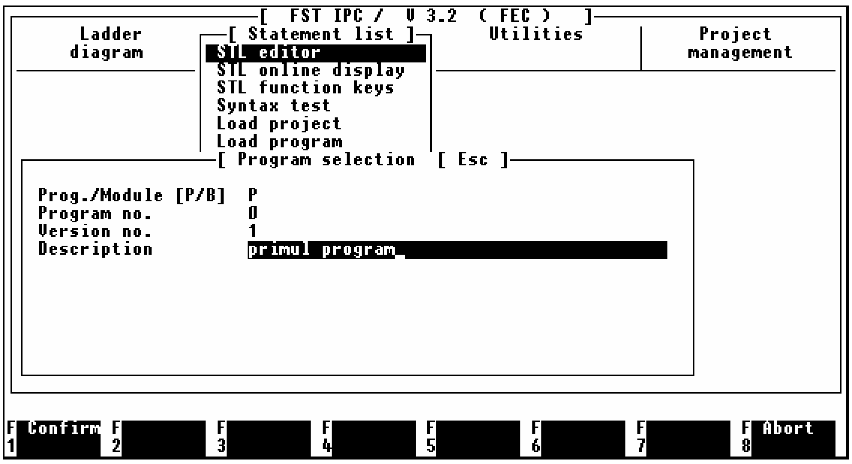


Fig. 8. *Crearea unui program STL.*

La selectarea primei opțiuni din submeniu, *STL editor (Ladder editor)*, se deschide o fereastră de vizualizare a programelor. Dacă nu există nici un program în proiectul curent se va deschide fereastra prezentată în figura 8.

Pentru crearea unui program sunt necesare informațiile:

- în câmpul *Prog./Module* se selectează litera 'P', care este şi selecție implicită pentru crearea unui program principal. Litera 'B' se foloseşte pentru crearea unui program modul, (vezi Cap. 6).
- în câmpul *Program no.* se scrie numărul programului. Numărul maxim de programe admise în lucrul cu FEC-ul este de 64. La pornirea FEC-ului sau la comutarea START/STOP, primul program care se va executa va fi P0. De aceea, acesta trebuie să existe întotdeauna.
- în câmpul *Version no.* se scrie versiunea programului. Acelaşi program, de exemplu P2, poate avea mai multe versiuni: P2.0, P2.1, P2.2. În memoria FEC-ului se va

încărca întotdeauna o singură versiune a unui program. Prin incrementarea numărului de versiune al unui program se obține automat o copie a vechiului program. Această copie poate fi modificată şi adaptată pentru noile cerințe ale aplicației.

- câmpul *Description* cuprinde o descriere sumară a programului, dată de utilizator.

Dacă există deja programe STL (LDR) în cadrul proiectului curent, la selectarea opțiunii *STL editor (Ladder editor)* se deschide o fereastră care conține o listă cu programele create, fig. 9. În continuare se apasă tasta F1 pentru crearea unui nou program.

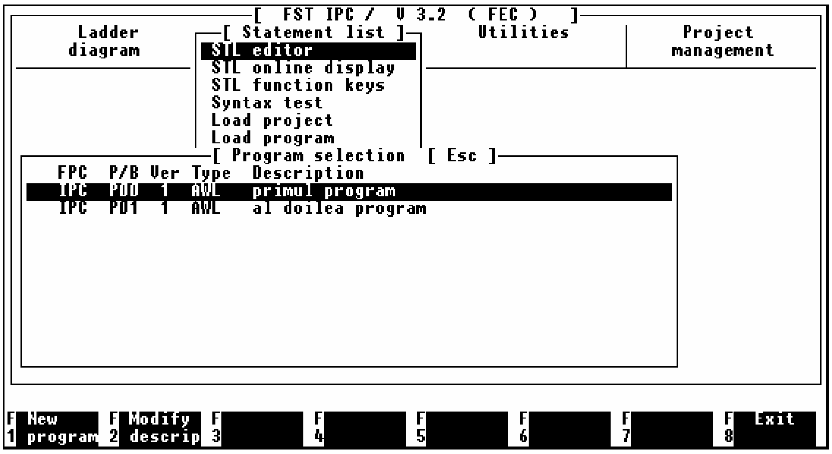


Fig. 9*. Lista programelor STL existente în proiectul curent.*

Odată creat un program se intră automat în fereastra de editare a programelor. În figura 10 este prezentată editarea unui mic program STL. Din fereastra de editare a programelor se poate ieşii oricând prin apăsarea tastei F8 şi după selectarea opțiunii de salvare sau nu a modificărilor efectuate în program. Pentru salvare se alege opțiunea *Save and quit editor*.

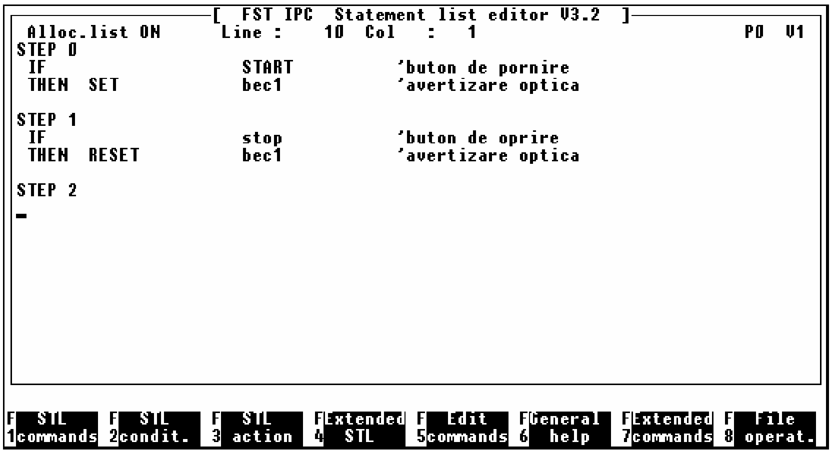


Fig. 10*. Fereastra de editare a programelor STL.*

Dacă utilizatorul doreşte ştergerea unui program, aceasta se poate face cu opțiunea *Delete Program* din submeniul *Project management*. Va apărea lista cu toate programele cuprinse în proiectul curent. Se selectează un program cu tastele direcționale (↑, ↓) după care se apasă tasta ENTER. Pentru confirmarea comenzii se va selecta *Y (Yes)* la întrebarea *Y/N?*.

UNIVERSITATEA TEHNICA GH. ASACHI IASI
FACULTATEA DE MECANICA
CATEDRA ORGANE DE MASINI SI MECATRONICA
Disciplina: Actionarea sistemelor mecatronice

LUCRARE DE LABORATOR NR. 3

MODULAR PRODUCTION SYSTEM (MPS)– ECHIPAMENT DE INSTRUIRE IN MECATRONICA

Programarea aplicațiilor în limbajul "*statement list*" (STL).

1. Scopul lucrarii:

- Dobandirea de cunostinte in programarea automatului programabil FEC-20-AC in limbajul STL

2. Introducere

La programarea în STL, programele sunt alcătuite prin scrierea de instrucțiuni *în modul text*. Ca orice limbaj de programare în mod text, limbajul STL foloseşte un set de cuvinte cheie. Ordinea de scriere a instrucțiunilor şi tipul acestora determină structura şi modul de funcționare a programelor.
Lucrarea de față explică cuvintele cheie şi structura limbajului STL aşa cum a fost implementat în automatele programabile Festo. Trebuie menționat că limbajele asemănătoare cu STL implementate de alte firme respectă aceleaşi principii şi aceeaşii logică de construcție a programelor chiar dacă folosesc alte cuvinte cheie.
Subiectele dezvoltate în acest capitol sunt: operanzi şi operatori STL, structura programelor STL, instrucțiuni STL, temporizatoare, numărătoare, module de programare.

2.1. Operanzi STL.

Operanzii sunt identificatori ai sistemului (intrări, ieşiri, timere, numărătoare, etc) şi reprezintă *numele* acestor resurse. În cadrul unui program utilizarea acestor operanzi este singurul mod de accesare a resurselor pe care le reprezintă.
În funcție de dimensiunea lor, există doua tipuri de operanzi:
- operanzi de un singur bit (SBO - single bit operands);
- operanzi de mai mulți biti (MBO - multibit operands) – în general 16 biți care formează un cuvânt (word).

Operanzii de un bit pot fi evaluați ca adevărați sau falşi (1 sau 0 logic). De asemenea, ei pot fi modificați din 0 logic în 1 logic sau invers. Evaluarea şi modificarea operanzilor de un bit se realizează cu ajutorul câtorva instrucțiuni specifice ce vor fi prezentate în acest capitol. În timpul interogării şi modificării operanzilor de un bit, aceştia se încarcă într-un registru special de un bit al procesorului, numit: Single Bit Accumulator (SBA).
Operanzii multibit sunt acele resurse care se pot accesa într-un octet (8 biți) sau cuvânt (word, 16 biți) şi care reprezintă numere întregi. Ei pot lua valori în domeniile:
- $0 \div 255$, pentru un octet (8 biți);
- $0 \div 65535$, pentru un cuvânt (16 biți) reprezentat ca număr intreg fără semn;
- $-32767 \div 32767$, pentru un cuvânt (16 biți) reprezentat ca întreg cu semn.

Valoarea operanzilor MBO poate fi testată prin comparare (<, >, =, etc) cu valori constante sau cu alți operanzi multibit. Instrucțiuni STL permit modificarea conținutului unui operand MBO prin:
- scrierea în aceştia a unor valori constante sau a valorilor altor operanzi MBO;
- incrementarea (adunarea cu o unitate a valorii operandului);
- decrementarea (scăderea cu o unitate a valorii sale);
- manipulare a valorii MBO prin intermediul operatorilor multibit aritmetici sau a celor logici.

În momentul modificării, operanzii MBO sunt încărcați într-un registru special al procesorului numit:
MultiBit Accumulator (MBA).

În tabelul 1. sunt prezentati operanzii de un bit. În tabelul 2. sunt prezentați operanzii multibit.

Tabelul 1. Operanzi SBO (single bit operand)

Operand	**Forma STL**	**Sintaxa: n şi *m* reprezintă numere oarecare.**	**Parte a sentinței în care poate fi folosită: C* – condițională; E* – executivă.**
Intrare (Input, Eingang)	I	In.m	C
Iesire (Output, Ausgang)	O	On.m	C E
Flag sau bit de memorie (Flag, Merker)	F	Fn.m	C E
Numărător (Counter)	C	Cn	C E
Temporizator (Timer)	T	Tn	C E
Program (Program)	P	Pn	C E

* explicații sunt date în paragraful 6.4.

Tabelul 2. Operanzi MBO (multibit operands).

Operand	**Forma STL**	**Sintaxa: n reprezintă un număr oarecare.**	**Parte a sentinței în care poate fi folosită: C* – condițională; E* – executivă.**
Cuvânt de intrare	IW	Iwn	**C**
Cuvânt de ieşire	OW	Own	C E
Cuvânt de memorie	FW	FWn	C E
Cuvântul (valoarea) unui numarator	CW	CWn	C E
Cuvântul prestabilit al unui numarator	CP	CPn	C E
Cuvântul (valoarea) unui timer	TW	TWn	C E
Cuvântul prestabilit al unui timer	TP	TPn	C E
Unitate funcțională	FU	FUn	C E
Registru	R	Rn	C E

* explicații sunt date în paragraful 6.4.

Operanzii sunt accesibili din orice program al proiectului curent. În lista de alocare, care a fost descrisă in capitolul 5, se pot da nume simbolice tuturor operanzilor SBO sau MBO. Lista de alocare se poate alcătui şi pe parcursul editării programului.

6.3. Operatori de un bit şi multibit în limbajul STL.
Se numesc operatori acele simboluri care apar pe lângă operanzi, modificând sau folosind valoarea numerică sau logică a acestora. De exemplu simbolul '+' este operatorul de adunare şi se aplică între doi operanzi multibit, rezultatul fiind suma celor doi operanzi. Există operatori care se aplică operanzilor de un singur bit şi operatori care se aplică operanzilor multibit. În tabelul 6.3 sunt prezentați toți operatorii ce pot fi utilizați în construcția programelor STL. În coloana "Observații" se specifică dacă operatorii sunt aplicabili la operanzi de un bit (SBO) sau multibit (MBO).

Tabelul 6.3. Operatori STL.

Simbol	**Utilizare**	**Observații**
N	Not – negatie	SBO
V	Valoare zecimala; folosit la introducerea valorilor constante de către utilizator.	MBO
V$	Valoare hexazecimala; folosit la introducerea valorilor constante de către utilizator.	MBO
V%	Valoare binara; folosit la introducerea valorilor constante de către utilizator.	MBO
+	Adunarea a doi operanzi	MBO
-	Scăderea a doi operanzi	MBO
*	Înmulțirea a doi operanzi	MBO
/	Împărțirea a doi operanzi	MBO
<	Compararea a doi operanzi …mai mic decât	MBO
>	Compararea a doi operanzi …mai mare decât	MBO
=	Compararea a doi operanzi …egal cu	MBO
<>	Compararea a doi operanzi …diferit de	MBO
<=	Compararea a doi operanzi …mai mic sau egal ca	MBO
>=	Compararea a doi operanzi …mai mare sau egal ca	MBO
(…)	Parantezele sunt folosite pentru stabilirea priorităților de evaluare în expresii complexe	SBO, MBO

2.2. Structura unui program STL.
Pentru scrierea unui program în limbajul STL se poate alege între trei structuri posibile:
1) program cu paşi (STEP program);
2) program paralel;
3) program executiv.

Primul tip de program este cel mai general exemplu. Următoarele două sunt cazuri particulare obținute prin simplificarea celui dintâi.

1) Un **program cu paşi** poate conține până la 255 de paşi. Paşii programului sunt delimitați de instrucțiunea STEP, urmată (opțional) de un *nume* sau un *număr* dat de utilizator. Următorul pas începe la următoarea instrucțiune STEP. Numărul sau numele

pasului este folosit pentru claritatea programului şi pentru instrucțiuni de salt la pasul de interes din oricare parte a programului. Instrucțiunea de salt se va studia în paragraful 6.5.
Un pas al programului cuprinde una sau mai multe *sentințe*. O sentință completă conține:

- o clauză *condițională* formată din cuvântul cheie IF … urmat de o *expresie logică* simplă sau complexă care se va evalua ca fiind adevărată sau falsă. În expresia logică pot apărea operanzi SBO, operatori de comparație cu operanzi MBO, alte instrucțiuni.
- o clauză *executivă* formată din cuvântul cheie THEN … urmat de un set de instrucțiuni care se vor executa numai în cazul în care partea condițională a fost evaluată ca fiind adevărată;
- o clauză *executivă alternativă* formată din cuvântul cheie OTHRW… urmat de un set de instrucțiuni care se vor executa dacă expresia logică din partea condițională (IF) se evaluează ca fiind falsă. Instrucțiunea OTHRW poate să nu apară dacă utilizatorul nu doreşte prelucrări decât în clauza THEN.

De reținut:
Toate instrucțiunile din partea executivă, THEN, se execută numai dacă expresia logică care urmerază după cuvântul cheie IF, este adevărată.

Partea executivă cuprinde toate instrucțiunile care urmează după cuvântul cheie THEN şi până la întâlnirea unei unuia din cuvintele cheie IF, STEP sau OTHRW.

Exemplul 1: Program cu paşi (STEP program)

```
STEP init           ; pasul 'init' al programului
IF          I0.1    ; dacă există semnal la intrarea I0.1 (parte condițională)
   AND   N I0.2     ; ŞI dacă NU există semnal la intrarea I0.2 (parte condițională)
THEN                ; atunci (parte executivă)
     SET O0.1       ; atunci transmite semnal la ieşirea  O0.1 (parte executivă)
     SET O0.2       ; transmite semnal la ieşirea O0.2 (parte executivă)

STEP 1              ; pasul '1' al programului
THEN
    RESET O0.3      ; nu transmite semnal la ieşirea O0.3
IF…
THEN…
OTHRW…
…
```

În prima sentință din cadrul unui pas (imediat după STEP) se poate omite partea condițională şi se poate începe direct cu o instrucțiune THEN, aceasta fiind o *sentință incompletă*. Instrucțiunile din această parte a programului şi până la întâlnirea unuia din cuvintele cheie IF şi STEP, se vor executa întotdeauna.
Un program STEP este executat pas cu pas. Programul va trece la pasul următor numai dacă, *la ultima sentință a pasului curent*, se execută:

- fie instrucțiunile care urmează după THEN (*partea condițională este **adevărată***);
- fie pe cele care urmează după OTHRW, dacă există (*partea condițională este **falsă***).

Dacă nu există clauza OTHRW şi partea condițională este falsă atunci programul reia (execută de la început) instrucțiunile pasului curent. În acest fel se 'aşteaptă' îndeplinirea ultimei condiții pentru trecerea la pasul următor.

De reținut:

În ultima sentință IF… THEN a pasului curent, se stabileşte dacă programul trece la pasul următor sau reia instrucțiunile pasului curent.

2) Un **program paralel** constă doar în una sau mai multe sentințe. Într-un astfel de program nu există paşi. În fapt, tot programul este echivalent cu *un pas* dintr-un program STEP. Prima sentința a acestui program poate fi una incompletă (începe direct cu THEN). Toate celelalte sentințe trebuie să fie complete.
Un program paralel este rulat ciclic (în continuu) până când acesta este dezactivat (oprit) prin comanda RESET Pn ('n' este numărul programului). Comanda de dezactivare se poate da din oricare alt program sau chiar din programul Pn. Ultima variantă cuprinde şi cazul în care se doreşte ca programul paralel să se execute o singură dată. În acest caz trebuie ca în ultima instrucțiune se existe o comandă RESET Pn. În exemplul 2, numele programului este P1. Se observă că la ultima sentință acesta se auto-dezactivează necondiționat.

Exemplul 2: Program paralel.

```
THEN             RESET F0.0 ; 'forțează' în 0 logic flagul F0.0
IF        N        I1.0   ; Dacă nu există semnal la intrarea I0.1
THEN             SET     O0.7 ; atunci transmite semnal la ieşirea O0.7.
IF                 I1.7   ; Dacă există semnal la intrarea I1.7
THEN             SET     O1.7 ; atunci transmite semnal la ieşirea O1.7
OTHRW     SET      F0.0 ; Altfel 'forțează' în 1 logic flagul F0.0
          RESET O1.7 ; şi nu transmite semnal la ieşirea O1.7.
...
IF                 F0.0   ; Dacă flagul F0.0 are valoarea logică 1
          AND      I1.0   ; şi dacă la intrarea I1.0 există semnal
THEN             SET     O1.0          ; atunci transmite semnal la ieşirea O1.0
          RESET P1    ; şi dezactivează programul curent, P1.
OTHRW     RESET P1    ; Altfel dezactivează programul curent, P1.
```

De reținut:
Un program paralel NU conține cuvântul cheie STEP.

3) **Programul executiv** este echivalent cu o sentință incompletă dintr-un program paralel. Diferența consistă în faptul că nu există o introducere cu cuvântul cheie THEN. Instrucțiunile dintr-un program executiv se execută în totalitate deoarece nu există o clauză condițională. Dacă se va introduce o clauză IF în interiorul programului va rezulta o eroare de sintaxă la compilarea programului. Exemplul 3 este un program executiv.

Exemplul 3: Program executiv.

```
     SET F0.0      ; setează flagul F0.0
     RESET O1.0          ; nu transmite semnal la ieşirea O1.0
     LOAD V50    ; încarcă în acumulatorul MBA numărul zecimal 50
     TO FW7       ; încarcă din acumulatorul MBA în cuvântul memoriei F7, valoarea
zecimală 50
     ……           ; alte instrucțiuni
```

De reținut: *Un program executiv NU conține cuvintele cheie STEP, IF şi THEN.*

2.3. Instrucțiuni STL.

Limbajul STL oferă un set de instrucțiuni care permit scrierea uşoară a programelor destinate a controla automatizări simple sau complexe. În tabelul 44. sunt prezentate, împreună cu o scurtă descriere, toate instrucțiunile limbajului STL implementat pe automatele programabile FESTO.

Tabelul 4. Instrucțiunile limbajului STL.

Instrucțiune		**Semnificație**
1	AND	Execută funcția ŞI logic între doi operanzi (de un bit sau multibit)
2	BID	Converteşte conținutul acumulatorului multibit din cod binar în cod BCD
3	CFM n	Începe execuția sau inițializează un modul functional (rutină de sistem standard)
4	CMP n	Începe execuția unui modul de program (subrutină sau funcție de bibliotecă)
5	CPL	Complementul lui 2 al acumulatorului multibit (echivalent cu o înmultire cu -1)
6	DEC	Decrementează un operand sau acumulatorul multibit
7	DEB	Converteşte conținutul acumulatorului multibit din cod BCD în cod binar
8	EXOR	Execută funcția logică SAU Exclusiv între doi operanzi (de un bit sau multibit)
9	IF	*Cuvant cheie* ce marcchează începutul părții condiționale a unei sentințe
10	INC	Incrementează un operand sau acumulatorul multibit
11	INV	Produce complementul față de 1 al acumulatorului multibit
12	JMP TO xx	Execută un salt la pasul cu numele (sau numărul): xx
13	LOAD xx	Încarcă operandul xx de un singur bit (sau multibit) în acumulatorul de un singur bit (sau multibit)
14	NOP	Instructiune specială, întotdeauna adevarată în partea conditionala. Este utilizată şi în partea executivă a unei sentințe unde execuția ei înseamnă: 'nu face nimic!'
15	OR	Executa funcția logică SAU între doi operanzi (de un bit sau multibit)
16	OTHRW	Oferă posibilitatea de a continua un program dacă partea condițională a unei instrucțiuni nu este adevărată
17	PSE	Program Section End. Reia programul sau pasul de la prima instrucțiune. Această instrucțiune nu este utilizabilă pentru FEC.
18	RESET	Schimbă valoarea unui operand de un bit. Valoarea finală este totdeauna 0 logic.
19	ROL	Roteşte la stânga cu o poziție toți biții conținuți de acumulatorul multibit. Bitul cel mai semnificativ (MSB) trece în cel mai putin semnificativ (LSB)
20	ROR	Roteşte la dreapta cu o poziție toți biții conținuți de acumulatorul multibit. Bitul cel mai puțin semnificativ (LSB) trece în cel mai semnificativ (MSB)
21	SET	Schimbă valoarea unui operand de un bit. Valoarea finală este todeauna 1 logic.
22	SHIFT	Execută o interschimbare între un operand de un singur bit şi acumulatorul de un singur bit (SBA)
23	SHL	Translatează la stânga cu o poziție toți biții conținuți de acumulatorul multibit. Bitul cel mai semnificativ (MSB) este pierdut iar cel mai puțin semnificativ (LSB) este resetat la 0 logic

24	SHR	Translatează la dreapta cu o poziție toți biții conținuți de acumulatorul multibit. Bitul cel mai puțin semnificativ (LSB) este pierdut iar cel mai semnificativ (MSB) este resetat la 0 logic
25	SWAP	Schimbă între ei octeții superior şi inferior ai acumulatorului multibit (MBA)
26	TO	Utilizată cu instrucțiunea LOAD pentru a specifica un operand destinație
27	THEN	*Cuvant cheie* ce indică începutul părții executive a unei instrucțiuni
28	WITH	Utilizat pentru a furniza parametrii modulelor CFM sau CMP (în cazurile în care acestea cer parametri)

În continuarea acestui paragraf se vor detalia cele mai folosite instrucțiuni şi se vor da exemple de utilizare a lor.

Instrucțiunile **SET** şi **RESET** sunt comenzi date operanzilor SBO pentru activare respectiv dezactivare. Comenzile, aplicate asupra aceluiaşi operator, se anulează una pe cealaltă. În tabelul 5. sunt explicate efectele comenzilor asupra operanzilor de un bit.

Tabelul 5. Acțiunea instrucțiunilor SET şi RESET asupra operanzilor SBO

Operand	**Sintaxa**	**Efect**
Ieşire	SET O1.6	Activează ieşirea 1.6
	RESET O1.6	Dezactivează ieşirea 1.6
Flag	SET F2.1	Forțează starea memoriei F2.1 în 1 logic
	RESET F2.1	Forțează starea memoriei F2.1 în 0 logic
Counter	SET C2	1) CW2 este încărcat cu valoarea 0 2) Bitul C2 este activat (devine 1 logic)
	RESET C2	1) Bitul C2 este dezactivat (devine 0 logic) 2) CW2 rămâne neschimbat
Timer	SET T5	1) Valoarea TP este încărcată în TW 2) Bitul T5 este activat (devine 1 logic)
	RESET T5	Bitul T5 este dezactivat (devine 0 logic)
Program	SET P1	Programul P1 este activat şi rulează de la început
	RESET P1	Programul P1 este dezactivat şi nu se mai execută

Instrucțiunea **LOAD… TO…** este deosebit de folosită în lucrul cu operanzi multibit. Ea se poate aplica şi la operanzi de un bit. Aplicată unui operand multibit, instrucțiunea LOAD are ca efect încărcarea valorii operandului respectiv în acumulatorul multibit (MBA). Dacă se aplică asupra unui operand SBO, valoarea logică a acestuia se va copia în acumulatorul de un bit (SBA).
Scopul instrucțiunii este de a permite efectuarea de operații logice sau matematice. De asemenea, acumulatoarul poate fi un pas intermediar pentru transferul unor valori între diferiți operanzi. Partea LOAD …*sursă*, încarcă în acumulator (SBA sau MBA) operandul sursă. Partea TO …*destinație*, descarcă valoarea din acumulator în operandul destinație. Între operațiile de încărcare şi descărcare se pot executa diverse alte operații asupra conținutului acumulatorului. Exemplele care urmează sunt semnificative.

a) Exemple pentru operanzi de un bit (SBO).

```
1)    LOAD     I1.0   ; încarcă în SBA starea logică a intrării I1.0
      TO       O1.0   ; copiază această stare la ieşirea O1.0
```

```
2)      LOAD        I1.0   ; încarcă în SBA starea logică a intrării I1.0
           AND   N I1.1   ; operație logică între SBA şi I1.0 negat. Rezultatul se înregistrează în SBA.
        TO          O0.1   ; copiază SBA la ieşirea O1.0
```

Exemplele de mai sus sunt instrucțiuni STL valabile dar se pot înlocui cu instrucțiuni de mai jos, care sunt mai uzuale:

```
1)      IF                 I1.0
        THEN    SET        O1.0
        OTHRW   RESET      O1.0

2)      IF                 I1.0
             AND   N       I1.0
        THEN    SET        O0.1
        OTHRW   RESET      O0.1
```

b) Exemple pentru operanzi multibit (MBO).

```
1)      LOAD        V100   ; încarcă în acumulator valoarea zecimală 100
        TO          TP1    ; copiază valoarea 100 în cuvântul Timer Preselect al temporizatorului 1
        TO          R10    ; copiază aceeaşi valoare 100 în registrul 10.

2)      LOAD        IW1    ; încarcă în acumulator conținutul cuvântului de intrare IW1
             AND    V$0F         ; execută funcția SI multibit cu valoarea hexazecimală 0F
        TO          OW0    ; copiază rezultatul în cuvântul de ieşire OW0
```

În exemplul 2 s-au păstrat numai intrările din primii 4 biți ai cuvântului de intrare IW1. Procedeul se numeşte ‘mascare’. Masca este valoarea hexazecimală 0f care în binar se scrie 00001111. După operația logică SI, pot fi diferiți de zero numai primii 4 biți.

3) Dacă un automat programabil are 4 cuvinte de ieşire de 16 biți fiecare: OW0, OW1, OW2, OW3, resetarea tuturor ieşirilor prin metode uzuale (RESET OW0.1, RESET OW0.2, … OW4.15) ar necesita 64 de linii de cod STL. Utilizând instrucțiunea LOAD această operație se poate face în 5 linii de cod, conform exemplului de mai jos.

```
        LOAD  V0
        TO    OW0
        TO    OW1
        TO    OW2
        TO    OW3
```

Observație:Instrucțiunea LOAD este una dintre cele mai ‘puternice’ instrucțiuni din limbajul STL. Trebuie reținut că o instrucțiune LOAD… ‘pregăteşte’ sistemul pentru următoarele instrucțiuni. Dintre instrucțiunile care, de regulă, se folosesc după LOAD…, se amintesc: SHL, SHR, ROL, ROR, SWAP, AND, OR, EXOR, etc.

Instrucțiunea **JMP TO…** furnizează un mijloc eficient de a influența cursul execuției programului. Prin utilizarea instrucțiunii JMP TO.. se poate ocoli regula ca ultima sentință dintr-un pas să fie adevărată pentru ca programul să treacă la pasul următor. Instrucțiunea

JMP TO.. poate fi utilizată pentru a stabili o *prioritate* de execuție a sentințelor dintr-un anumit pas al programului. Exemplul următor foloseşte instrucțiunea JMP TO.. pentru tratarea situației în care s-a apăsat butonul EMERGENCY STOP. În funcționare normală acest buton este normal închis.

```
STEP 20
………                        ; alte sentințe în pasul 20
IF          N      I1.1      ; Dacă s-a apăsat butonul EMERGENCY STOP
THEN        LOAD V0          ; incarcă valoarea 0 în acumulator
            TO     OW0       ; resetează toate ieşirile (uzual pentru cazuri de urgență)
…….    JMP TO      80        ; alte sentințe în pasul 20
…………….                      ; alți paşi ai programului
STEP 80
IF                 I1.1      ; Dacă s-a apăsat butonul de oprire de urgență aşteaptă în
       AND         I2.1      ; acest loc până se va apăsa butonul de reset (I2.1) a
THEN   JMP TO      20        ; instalației şi atunci se va continua programul de la pasul 20
```

Următorul exemplu de utilizare a instrucțiunii JMP TO… ilustrează situația în care un operator poate selecta o opțiune din 3 posibilități.

```
STEP 40
IF                 I1.1      ; Dacă intrarea 1.1 este singura activă
       AND   N     I1.2      ;
       AND   N     I1.3      ;
THEN  JMP TO       100       ; execută salt la pasul 100

IF           N     I1.1      ;
       AND         I1.2      ; Dacă intrarea 1.2 este singura activă
       AND   N     I1.3      ;
THEN  JMP TO       150       ; execută salt la pasul 150
IF           N     I1.1      ;
       AND   N     I1.2      ;
       AND         I1.3      ; Dacă intrarea 1.3 este singura activă
THEN  JMP TO       200       ; execută salt la pasul 200
```

Prin ordonarea atentă a mai multor sentințe într-un pas, împreună cu folosirea instrucțiunii JMP TO…, se pot stabili priorități de execuție a programului. În exemplul ce urmează se presupune că toți paşii până la 50 conțin instrucțiuni pentru procese de automatizare iar de la pasul 50 până la pasul 60, programul verifică intrările I1.1, I1.2, I1.3 şi aşteaptă până la apariția uneia din acestea. Numai una din cele 3 posibile intrări va fi tratată şi, dacă există mai multe intrări la un moment dat, intrarea I1.1 va avea cea mai mare prioritate.

```
STEP 60
IF           N     I1.1      ; Dacă nici o intrare
       AND   N     I1.2      ; din cele trei
       AND   N     I1.3      ; nu este activă
```

```
            THEN JMP TO     60    ; atunci reia execuția pasului 60 (buclă de
aşteptare)

            IF              I1.1  ; Dacă intrarea 1.1 este activă
            THEN JMP TO     100   ; execută salt la pasul 100

            IF              I1.2  ; Dacă intrarea 1.2 este activă
            THEN JMP TO     150   ; execută salt la pasul 150

            IF          N   I1.1  ;
                  AND   N   I1.2  ;
                  AND       I1.3  ; Dacă intrarea 1.3 este activă
            THEN            NOP   ; atunci nu executa nimic şi treci mai departe
(pasul următor)
```

Instrucțiunea **NOP** înseamnă 'nici o operație' (NO Operation) şi oricât ar părea de ciudată este deseori utilă în programele STL. Consecințele utilizării acestei instrucțiuni depind de locul unde este utilizată într-o sentință. În partea condițională instrucțiunea NOP va fi întotdeauna adevărată (1 logic) şi deci instrucțiunile din partea executivă se vor executa.

```
            STEP 45
            IF              NOP   ; întotdeauna adevărat (1 logic)
            THEN        SET    T6 ; atunci startează temporizatorul T6
                        RESET F1.1; resetează bitul 1 al flag-ului FW1
```

Dacă un pas al unui program conține mai multe sentințe care trebuie să fie prelucrate continuu, instrucțiunea NOP poate fi utilizată pentru controlul cursului execuției programului.

```
            STEP 11
            IF              I1.4  ; Dacă intrarea 1.4 este activă
            THEN        SET T4    ; atunci startează temporizatorul 4

            IF              I3.0  ; Dacă este activă intrarea 3.0 (START MANUAL)
            THEN        SET O1.6  ; atunci porneşte motorul (ieşirea 1.6)
            OTHRW  RESET    O1.6  ; altfel opreşte motorul

            IF              T4    ; Dacă temporizatorul 4 este pornit
                    AND     O1.6  ; şi motorul este pornit
            THEN        INC CW3   ; incrementează numărătorul 3

            IF          N   I2.2  ; Dacă s-a apăsat butonul EMERGENCY STOP
            THEN  JMP TO    90    ; atunci execută salt la pasul de tratare a acestei
situații

            IF              NOP   ; Întotdeauna
            THEN  JMP TO    11    ; continuă de la începutul pasului curent

            STEP 90
```

```
        IF                  I2.2   ; Dacă butonul EMERGENCY STOP a fost reactivat
                AND         I3.3   ; şi butonul RESET a fost apăsat (se aşteaptă aceste condiţii)
        THEN  JMP TO        11     ; atunci continuă 'scanarea' pasului 11
```

Când este folosită în partea executivă a unei sentinţe instrucţiunea NOP este echivalentă cu… *'nu face nimic'*. Deşi acest lucru pare să nu aibă valoare practică, instrucţiunea este utilă în cazurile când programul trebuie să *aştepte* îndeplinirea unui set de condiţii înainte de a se merge mai departe. Exemplul următor relevă această situaţie.

```
        STEP 60
        IF                  I1.5   ; Dacă intrarea 1.5 este activă
            AND             T7     ; şi temporizatorul 7 merge
            AND   N         C2     ; şi numărătorul 2 a terminat de numărat
        THEN                NOP    ; atunci condiţiile de mai sus sunt satisfăcute şi se poate
                                   ; merge mai departe
```

Instrucţiunile **INC** şi **DEC** se folosesc pentru incrementarea, respectiv decrementarea cu 1 a operandului multibit la care sunt aplicate. Spre deosebire de alte operaţii aritmetice, nu este nevoie de încărcarea operandului în acumulator înaintea operaţiei propriu-zise. Instrucţiunile INC şi DEC se pot utiliza cu orice operand multibit dar, de regulă, ele se utilizează pentru actualizarea numărătoarelor. Aplicarea ambelor instrucţiuni asupra unui operator multibit este exemplificată mai jos.

Pe o linie automată de încărcat sticle, intrarea I1.3 este activată de fiecare dată când o sticlă trece prin dreptul staţiei de numărare. Numărul total de sticle este memorat în registrul R9. Se poate întâmpla ca o sticlă să nu fie umplută în totalitate şi acest lucru se sesizează cu senzorul I3.6 plasat după staţia de numărare. În cazul acesta numărul total de sticle memorat în registrul R9 trebuie scăzut cu 1 iar sticla respectivă va fi reintrodusă în circuitul de umplere. Programul STL care realizează această automatizare este prezentat în continuare.

```
    IF                  I1.3   ; intrarea 1.3 sesizează toate sticlele care trec prin staţia de numărare
    THEN        INC     R9     ; şi actualizează numărul acestora în registrul R9
    IF                  I2.2   ; Dacă o sticlă a ajuns la staţia de testare a nivelului de umplere
        AND     N       I3.6   ; şi sticla nu este plină
    THEN        DEC     R9     ; atunci se reduce cu 1 numărul total al sticlelor memorat în R9
                SET     O2.1   ; şi se reintroduce sticla în circuitul de umplere
```

Instrucţiunea INC R9 este echivalentă cu instrucţiunile:

```
    LOAD        R9
        +V1
    TO          R9
```

ceeace accentuează observaţia despre flexibilitatea instrucţiunii LOAD…TO…

Instrucțiunea **AND** este folosită :

- pentru operația logică SI între doi sau mai mulți operanzi SBO sau MBO în partea condițională a unei sentințe;
- pentru evaluarea funcției SI logic între doi operanzi MBO fie în partea condițională fie în cea executivă a unei sentințe

Exemple:

1) operanzi SBO.

```
        IF          I1.1   ; daca intrarea 1.1 are semnal
             AND    T6     ; şi temporizatorul T6 este activat
        THEN SET O1.5      ; atunci transmite semnal la ieşirea 1.5
```

2) operanzi MBO

a) utilizare în partea condițională

```
        IF     (     R6    ; dacă operandul multibit obținut prin funcția ŞI logic
             AND     R7 )  ; între regiştrii 6 şi 7
             =       V34   ; este egal cu 34 (valoare zecimală)
        THEN ….            ; atunci …. (partea executivă)
```

b) utilizare în partea executivă

```
        IF     ….                ; dacă… (parte condițională)
        THEN  LOAD (       R38   ; atunci încarcă în acumulatorul multibit operandul
             AND           R45)  ; obținut cu funcția SI logic între regiştrii 38 şi 45
             TO            R17   ; descarcă conținutul acumulatorului în registrul
17.
```

Instrucțiunile **OR** şi **EXOR** au acelaşi regim de utilizare ca şi instrucțiunea AND. În continuare se va da un exemplu de aplicare a funcției EXOR (Sau Exclusiv).

Exemplu:

Pe o stație de umplere în serii de 8 sticle, există 8 poziții de umplere. Sticlele sosesc la această stație transportate pe o bandă rulantă. Când o sticlă ajunge în poziția 8 linia se opreşte şi sticlele prezente sunt umplute. În fiecare poziție de umplere (în afară de poziția 8) este posibil ca la un moment dat să existe sau nu o sticlă. Când la stație s-au terminat de umplut sticlele prezente ciclul se reia pentru o nouă serie.

Intrările I0.0 … I0.7 sunt conectate la senzorii de prezență a sticlelor în pozițiile de umplere. Intrarile I1.0…I1.7 sunt conectate la senzorii de umplere. Ieşirile O0.0…O0.7 controlează dozatoarele de lichid. Ieşirea O1.0, când este setată, închide o poartă pentru a imobiliza sticlele în timpul umplerii.

```
STEP 10
IF           N    O1.0  ; Dacă sticlele nu sunt oprite
     AND          I0.7  ; şi o sticlă există la ultima poziție (poziția I0.7)
THEN SET          O1.0  ; opreşte sticlele din mişcare
LOAD         (    IW0   ; înregistrează care sticle sunt prezente
     EXOR         IW1)  ; şi nu sunt umplute
TO                OW0   ; comandă umplerea sticlelor.
IF           (    OW0   ; Dacă toate ieşirile
             =    V0 )  ; sunt oprite
     AND          O1.0  ; şi linia este oprită pentru umplere
```

```
THEN RESET      O1.0  ; atunci reporneşte banda pentru următorul ciclu
        JMP TO  10    ; şi reia programul de la pasul 10
```

Instrucțiunile **SHL** (SHift Left) şi **SHR** (SHift Right) sunt instrucțiuni care deplasează toți biții acumulatorului multibit cu o poziție spre stânga (în cazul SHL) sau spre dreapta (în cazul SHR). Bitul care dispare după această mutare va fi pierdut. Bitul care apare va fi 0 logic. În tabelul 6.6. se dă un exemplu de folosire al celor două comenzi.

Tabelul 6.6. Efecte ale instrucțiuniloe SHL şi SHR

Rezultat	**Operație**
1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1	**LOAD R10**
1 ← 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 **0**	SHL (prima)
0 ← 1 0 1 1 1 1 0 0 0 0 1 0 0 1 **0 0**	SHL (a doua)
1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1	LOAD R10
0 1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 → 1	SHR (prima)
0 0 1 0 1 0 1 1 1 1 0 0 0 0 1 0 → 0	SHR (a doua)

Instrucțiunea SHL poate fi folosită pentru înmulțirea cu 2 a conținutului acumulatorului. Utilizatorul trebuie să verifice înainte că bitul 16 nu este 1 logic. Altfel valoarea obținută nu mai este rezultatul înmulțirii cu 2 şi se obține o eroare numită 'overflow'.
În acelaşi mod pentru împărțirea valorii acumulatorului multibit cu 2 se poate folosi instrucțiunea SHR. Rezultatul este un număr întreg chiar dacă valoarea inițială a fost un număr impar.
Exemplul următor prezintă o secțiune de program STL pentru automatizarea unei linii de asamblare. Instrucțiunea SHL se foloseşte pentru simplificare programării prin utilizarea repetată a acesteia asupra aceluiaşi operand multibit. Metoda se numeşte "Shift Register".

Exemplu:O linie de asamblare de cartuşe cu bandă pentru imprimante este compusă din 10 posturi de lucru. Procesul începe în postul numărul 1 unde o carcasă de cartuş este plasată pe linia de asamblare. La postul numărul 10 cartuşul complet este descărcat într-o maşină de ambalat. La fiecare post de lucru (1...10), după teminarea operației de asamblare corespunzătoare, este efectuată şi o verificare a calității. Piesele cu defecte sunt imediat scoase de pe linia de asamblare. În plus, când maşina se porneşte dimineața (după ce s-a oprit seara), numai posturile de lucru în care existau componente bune (la oprire) trebuie să reia lucrul. Fiecare post de lucru conține senzori pentru asigurarea poziționării corespunzătoare a pieselor înaintea operației de asamblare.

Ca operand multibit se foloseşte cuvântul de memorie (flag) FW1. În primii 10 biți ai cuvântului FW1 vor fi 1 logic numai acei biți corespunzători posturilor de lucru valide.

STEP 40

```
IF              F1.1  ; Dacă în postul 1 există o piesă bună
    AND   N     T1    ; şi operația de asamblare s-a terminat
    AND   N     I2.1  ; iar senzorul de calitate este neactivat (piesa nu este bună)
```

```
THEN RESET        F1.1   ; atunci dezactivează bitul 1 al cuvântului FW1.

IF                F1.2   ; la fel ca mai sus pentru postul 2
      AND   N     T1     ; …
      AND   N     I2.2   ; …
THEN RESET        F1.2   ; …
………………………………
IF                F1.10  ; la fel ca mai sus pentru postul 10
      AND   N     T1     ; …
      AND   N     I2.10  ; …
THEN RESET        F1.10  ; …

IF                T1     ; Dacă operaţia s-a terminat
THEN SET          O1.1   ; atunci indexează linia de asamblare
```

STEP 50

```
IF          N     I2.0   ; linia de asamblare este încă în mişcare pentru indexare
THEN LOAD         FW1    ; încarcă cuvântul FW1 în acumulator
      SHL                ; deplasează la stânga conţinutul acumulatorului
      +     V1           ; setează primul bit la 1 logic (în postul 1 va fi o piesă nouă)
      TO          FW1    ; şi copiază rezultatul în acelaşi cuvânt FW1
```

STEP 60

```
IF                I2.0   ; Dacă linia a terminat mişcarea de indexare
THEN RESET        O1.1   ; atunci opreşte mişcarea ei
      JMP   TO    20     ; şi reia programul de la pasul 40.
```

Instrucţiunile **ROL** şi **ROR** rotesc conţinutul acumulatorului multibit spre stângă respectiv dreapta cu o poziţie. Utilizarea lor este asemănătoare cu cea a instrucţiunilor SHL şi SHR. Tabelul 6.7. explică efectul aplicării acestor instrucţiuni asupra MBA.

Tabelul 6.7. Efecte ale instrucţiuniloe ROL şi ROR

Rezultat	**Operaţie**
1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1	**LOAD R10**
1 ← 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 **1**	ROL (prima)
0 ← 1 0 1 1 1 1 0 0 0 0 1 0 0 1 **1 0**	ROL (a doua)
1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1	LOAD R10
1 1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 → 1	ROR (prima)
0 1 1 0 1 0 1 1 1 1 0 0 0 0 1 0 → 0	ROR (a doua)

Instrucţiunile **CPL** şi **INV** sunt asemănătoare deşi se aplică în situaţii diferite. Instrucţiunea CPL se foloseşte pentru operaţia de complement faţă de 2 a conţinutului acumulatorului multibit (MBA). Reprezentarea în complement faţă de 2 este o tehnică uzuală în aritmetica binară. Prin această tehnică se reprezenta digital numerele negative. Când CPL se aplică

la numere întregi cu semn se obține numărul înmulțit cu (–1). De exemplu complementul față de 2 al numărului 10 este –10. Aceasta din urmă este principala utilizarea a instrucțiunii CPL. Tabelul 6.8. arată modul de obținere a unui complement fată de 2 aplicat la un operand de 16 biți.

Tabelul 6.8. Rezultate obținute cu instrucțiunile CPL şi INV

Operand	**Operație**
0 0 0 1 0 1 1 1 1 0 0 1 1 1 0 1	MBO
1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1	CPL (= INV + 1)
1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 0	INV (= CPL – 1)
Mod de obținere a rezultatului	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 – (= 2^{16} – 1) <u>0 0 0 1 0 1 1 1 1 0 0 1 1 1 0 1</u> (operand MBO) 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 0 + rezultat INV <u>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1</u> (+ 1) 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 rezultat CPL	Formula de obținere a complementului fată de 2 al unui MBO (16 biți) este: CPL(MBO) = 2^{16} – MBO 2^{16} = 1 0000 0000 0000 0000 = 1111 1111 1111 1111 + 0000 0000 0000 0001

Instrucțiunea INV inversează toți biții acumulatorului multibit (MBA). Când se aplică la intregi cu semn, operația este echivalentă cu înmulțirea numărului cu (–1) după care acesta se adună cu acelaşi (– 1). De asemenea instrucțiunea INV este echivalentă cu operația de negare bit cu bit a unui operand multibit. Următorul cod STL exemplifică utilizarea instrucțiunii INV.

O maşină de amestecat (mixer) are 16 puncte de lucru. Ciclul de lucru constă în alternarea în timp a perioadelor de scuturare cu cele de aşteptare a limpezirii produsului. La funcționarea normală a maşinii, muncitorii înlocuiesc aleator oricare din cele 16 containere de lucru. De aceea, la pornirea operației de scuturare, trebuiesc activate numai posturile de lucru care conțin containere. Pentru detectarea containerelor se folosesc senzori de proximitate.

```
STEP 10
IF          N   T1    ; Dacă ciclul "scuturare – aşteptare"  s-a încheiat
THEN LOAD       OW1   ; atunci încarcă în MBA starea ieşirilor
     INV              ; inversează rezultatul (stațiile care au stat, acum vor lucra)
     AND        IW1   ; verifică prezența containerelor ( dar numai cele cu containere)
     TO         OW1   ; noua comandă de lucru a stațiilor
     SET        T1    ; activează temporizatorul 1
STEP 20
IF          N   T1    ; aşteaptă trecerea timpului necesar completării procesului
THEN JMP TO     10    ; după care se reia pasul 10.
```

Instrucțiunea **SHIFT** execută o schimbare între valorile logice ale unui operand de un bit (SBO) şi acumulatorul de un bit (SBA). Instrucțiunea este folosită pentru operații de

manipulare la nivel de bit. În acest fel se poate modifica orice bit dintr-un operand multibit. Înainte de folosirea instrucțiunii trebuie încărcat în acumulatorul de un bit (SBA) valoarea logică dorită.

În exemplul următor de fiecare dată când intrarea I1.0 este activată ieşirile O1.1 până la O1.4 trebuie actualizare în felul următor.
- Ieşirea O1.4 va lua valoarea anterioară a ieşirii O1.3;
- Ieşirea O1.3 va lua valoarea anterioară a ieşirii O1.2;
- Ieşirea O1.2 va lua valoarea anterioară a ieşirii O1.1;
- Ieşirea O1.1 va lua valoarea intrării I1.1;

```
STEP 10
IF                      I1.0  ; Dacă intrarea I1.0 este activă
THEN        LOAD        I1.1  ; atunci încarcă în SBA starea intrării I1.1
            TO          F0.0  ; această stare este memorată şi în flagul F0.0
      SHIFT             O1.1  ; în O1.1 va fi memorată starea lui I1.1 (în SBA se va
încărca O1.1)
      SHIFT             O1.2  ; în O1.2 va fi memorată starea lui O1.1 (în SBA se va
încărca O1.2)
      SHIFT             O1.3  ; în O1.3 va fi memorată starea lui O1.2 (în SBA se va
încărca O1.3)
      SHIFT             O1.4  ; în O1.4 va fi memorată starea lui O1.3 (în SBA se va
încărca O1.4)
STEP 20
IF          N           I1.0  ;
THEN   JMP TO           10    ;
```

Instrucțiunea **SWAP** se aseamănă cu instrucțiunea SHIFT dar se aplică pentru operanzi multibit. Efectul ei este de a inversa valorile celor doi octeți ai acumulatorului multibit MBA. Înainte de utilizare trebuie încărcat în acumulator un operand multibit. În tabelul 6.9. se dă un exemplu simplu din care se poate observa efectul acestei instrucțiuni.

Tabelul 6.9. Aplicație a instrucțiunii SWAP.

0000 0000 1111 1111	LOAD MBO
1111 1111 0000 0000	SWAP
1111 1111 0000 0000	TO MBO

Instrucțiunile **BID** şi **DEB** sunt folosite pentru trecere din sistemul binar în cel binar codat zecimal (BID) şi invers (DEB). Sistemul de numerație binar este cel utilizat în mod obişnuit în automatul programabil. Sistemul binar codat zecimal este folosit pentru comunicarea cu aparate de afişare digitală. În acest din urmă sistem fiecare 4 biți reprezintă o cifră. Deci 16 biți pot reprezenta 4 cifre pe un afişor digital. Cu patru cifre se poate număra de la 0 la 9999. În tabelul 6.10. se prezintă corespondența între cifrele din baza 10 şi valorile lor în sistemul binar codat zecimal. În ambele cazuri operandul MBO trebuie încărcat în acumulator. Instrucțiunea BID transformă numărul binar într-un număr de 4 cifre, binar codat decimal. Instrucțiunea DEB transformă cele 4 cifre în număr binar.

Tabelul 6.10. Sistemul binar codat zecimal

0	1	2	3	4	5	6	7	8	9	Zecimal
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	**Binar**

2.4. Temporizatoare (Timers).

Temporizatoarele în automatele programabile FESTO sunt tipuri speciale de date constituite din trei operanzi. Referindu-ne la timerul numărul 'n', aceştia sunt:

- **Tn** – Timer Status;
- **TPn** – Timer Preselect;
- **TWn** – Timer Word;

Pentru FEC sunt disponibile 256 de temporizatoare, numerotate de la 0 la 255.

Tn – Timer Status – este un operand de un bit (SBO) care reprezintă *starea timer-ului*. Acest operand poate fi interogat în orice moment pentru valoarea sa logică. De asemenea el poate fi activat şi dezactivat cu instrucțiunile SET Tn respectiv RESET Tn.

TPn – Timer Preselect – este un operand multibit (MBO) de un word (16 biți) care reprezintă *valoarea setată* de utilizator în unități de intervale de timp. Activarea sau dezactivarea temporizatorului Tn nu are influență asupra operandului TPn. Setarea unei valori în TPn se face explicit cu o instrucțiune de încărcare.

```
Exemplu:    LOAD        V200  ; încarcă valoarea zecimală 200 (2 secunde)
                  TO    TP2   ; în cuvântul Preselect al temporizatorului 2
```

La anumite automate programabile, intervalele de timp, numite şi baze de timp, pot fi configurate de utilizator (de exemplu 0,01s; 0,1 s; 1s; 10s). La automatul programabil FEC baza de timp este de o sutime de secundă (0,01s), acesta fiind cel mai mic interval de timp care se poate temporiza. Deoarece numărul maxim al unui word este de 65535 rezultă că cea mai mare durată de timp care poate fi obținută pentru un timer este de 655s, adică aproximativ 10 minute. Pentru durate de timp mai mari se pot folosi combinații între un timer şi un counter.

TWn – Timer Word – este un operand multibit (MBO) de un word (16 biți) care reprezintă *valoarea curentă* a timer-ului. Acest operand nu conține un număr stabil. La activarea temporizatorului valoarea TPn se încarcă automat în TWn. Acest număr va fi apoi decrementat (tot automat) la fiecare impuls de ceas. Când TWn ajunge la zero temporizatorul se dezactivează.

Utilizarea temporizatoarelor.

Înainte de a putea folosi un timer acesta trebuie inițializat, adică trebuie încărcată o valoare în Timer Preselect. Inițializarea trebuie făcută din nou numai dacă se doreşte modificarea duratei de timp de temporizare. Nu este necesar să se reinițializeze TP de fiecare dată când timer-ul este activat. Timer Preselect poate fi încărcat cu o valoare constantă sau cu conținutul unui operand multibit (regiştri, cuvânt de memorie sau de intrări, etc.)

La activarea unui timer cu instrucțiunea SET Tn, următoarele acțiuni se execută automat:

- se va activa timer-ul (Tn = 1 logic);
- se va încărca în TWn conținutul lui TPn după care;
- se va începe temporizarea.

Pe durata temporizării, la fiecare impuls de ceas, va fi decrementat operandul TWn. Când numărul TWn devine zero atunci temporizarea se încheie şi timerul se resetează, adică operandul Tn devine 0 logic.

Un timer se poate dezactiva la terminarea temporizării sau în mod explicit, cu comanda RESET Tn. În acest caz Timer Status, Tn, devine 0 logic. Dacă Tn era 0 logic în momentul comenzii de resetare atunci comanda nu are nici un efect.

2.5. Numărătoare (Counters).

Numărătoarele, ca şi temporizatoarele, sunt integrate în sistemul de operare al automatului programabil FEC. Numărătoarele pot fi utilizate pentru înregistrarea în timp a unui număr de evenimente (de exemplu numărul de piese care trec pe o bandă rulantă).

Pentru FEC sunt disponibili 256 de numărătoare, numerotate de la 0 la 255. Există două tipuri de numărătoare: incrementale şi decrementale. Tipul standard în STL este *tipul incremental* care se va studia în continuare.

Ca şi temporizatoarele, numărătoarele sunt structuri de date formate din trei operanzi. Pentru cazul general al unui numărător oarecare 'n', aceştia sunt:

- Cn – Counter Status ;
- CPn – Counter Preselect;
- CWn – Counter Word.

Cn – Counter Status – este un operand de un bit (SBO) în care se memorează starea numărătorului. În partea executivă a unei sentințe numărătorul poate fi activat sau dezactivat cu comenzile SET Cn şi RESET Cn. Operandul Cn poate fi interogat în orice parte condițională a programului.

CPn – Counter Preselect – este un operand multibit (MBO) de un cuvânt în care se încarcă *valoarea setată* de utilizator. Acest operand rămâne neschimbat în timpul numărării. Schimbarea valorii CPn se efectuează o singură dată, la inițializare. Operandul CPn poate fi inițializat cu o valoare constantă sau cu valoarea conținută în orice alt operand multibit (IWn, FWn, etc.). În cazul valorilor constante, limitele de variație ale acestora pot fi:

- V0 … V65535, întreg zecimal de 16 biți, fără semn;
- V– 32268 … V32267, întreg zecimal de 16 biți, cu semn;
- V$0000…V$FFFF, întreg reprezentat în baza 16 (hexazecimal);

CWn – Counter Word – este un operand multibit (MBO) de un cuvânt în care se găseşte *valoarea curentă* a numărătorului. Incrementarea numărătorului se face cu instrucțiunea 'INC CWn'.

Utilizarea numărătoarelor.

Înainte de utilizarea unui numărător acesta trebuie inițializat. Pentru numărătoare incrementale, la inițializare se încarcă valoarea 0 în CWn. Această operație se efectuează automat la activarea numărătorului. Pentru activarea unui numărător este suficientă comanda 'SET' urmată de operandul de stare al acestuia, 'Cn'. De fiecare dată când se activează un numărător se efectuează următoarele acțiuni:

- Valoarea operandului Counter Word, CWn, este încărcată cu valoarea 0;
- Bitul Couter Status, Cn, este setat la 1 logic.

Dacă numărătorul incremental Cn este deja activ şi se reactivează cu comanda 'SET Cn', atunci numărătorul este 'repornit' prin încărcarea valorii 0 în CWn.

Un numărător activat îşi menținea valoarea (Cn = 1 logic) până la apariția uneia din situațiile de mai jos:

- numărătorul se resetează cu comandă RESET Cn;
- după incrementări succesive ale operandului CWn, acesta devine egal cu valoarea setată: CWn = CPn;

După apariția oricăreia din situațiile de mai sus numărătorul se dezactivează (Cn=0 logic).

2.6. Elemente avansate de programare STL. Module software.

În plus față de programele concepute de utilizator se pot folosi module software care sunt de două feluri:

- module de funcții;
- module program;

Un modul software poate fi apelat din orice parte executivă a oricărui program STL.

2.6.1. Module de funcții create de constructor (CFM).

Modulele de funcții, CFM, sunt mici programe utilizate pentru a efectua prelucrări de date de complexitate mică sau medie. Deoarece sunt utile şi des folosite de utilizatori, ele sunt furnizate de producător şi pot fi folosite imediat după instalarea softului. La instalare mediului FST-FEC, modulele de funcții sunt plasate în directorul FBLIB.FEC. Dacă se doreşte utilizarea lor, ele trebuiesc importate în proiectul curent cu ajutorul opțiunii *Import file* din meniul *Project management*. Numai în acest fel ele sunt disponibile pentru apelare în cadrul programelor. După selectarea funcțiilor modul din directorul FBLIB.FEC acestora li se atribuie un număr de către utilizator.

O funcție CFM este întotdeauna apelată din partea executivă a unei sentințe. În programul STL apelarea unei funcții modul se face cu instrucțiunea compusă:

```
CFM n
        WITH x1
        WITH x2
        …
```

unde 'n' este numărul dat de utilizator la importul fişierului iar 'x1', 'x2', sunt parametrii de intrare necesari funcției. Ca parametri se folosesc operanzi de diferite tipuri (intrări, ieşiri, memorie, numere constante, etc.).

2.6.2 Module program create de utilizator (CMP).

Modulele program, CMP, sunt subrutine create de utilizator în mediul FST-FEC. Sunt tratate în acelaşi fel ca programele obişnuite, fiind gestionate cu aceleaşi comenzi.

Pentru a crea un modul program trebuie selectată litera "B" în câmpul "Prog./Module [P/B]:" din fereastra de creare a unui nou program. Numărul modulului poate fi în domeniul 0…99. Editarea unui modul program este identică cu a unui program obişnuit. Totuşi există câteva detalii care trebuiesc reținute când se lucrează cu module program:

- nu se poate apela un CMP dintr-un alt CMP care conține paşi;
- programul care a apelat un CMP este oprit la instrucțiunea imediat următoare şi va aştepta până când execuția modulului program se va încheia;
- sunt permise instrucțiuni de salt, dar trebuiesc luate măsuri pentru evitarea formării buclelor infinite;

Procedura de scriere a instrucțiunii de apel a unui modul program este identică cu cea de la module de funcții cu observația că se înlocuieşte notația CFM cu CMP. Unui modul program i se pot plasa parametri în mod asemănător modulelor funcție, CFM. Aceasta se face cu instrucțiunea WITH urmată de o constantă sau de un operand multibit. Parametrii modulelor program sunt încărcați în unitățile speciale de funcții: FU32…FU38. În programul STL al modulului program este indicat să se folosească aceste funcții deoarece numai în acest fel modulul va fi aplicabil în orice situație. Dacă în modul se specifică un anumit operand absolut (de exemplu cuvântul intrărilor IW1) atunci modulul va putea fi folosit numai pentru o aplicație specifică şi nu mai are un caracter general (de exemplu pentru un alt automat programabil configurația intrărilor poate fi: IW0 sau IW7).

Apelarea unui program modul se va face întotdeauna din partea executivă a unei sentințe.

2.7. Verificarea corectitudinii programelor şi încărcarea unui proiect în AP.

Înainte de încărcarea programelor în automatul programabil acestea trebuiesc *compilate*. Prin compilare se înțelege transcrierea instrucțiunilor scrise în limbaj STL într-un limbaj special numit *cod maşină* pe care automatul îl poate înțelege. Pentru a simplifica munca utilizatorului, mediile de programare oferă unelte de verificare a corectitudinii scrierii programelor. Verificarea se poate face înainte de compilare sau la compilare. În cazul în care programele conțin erori şi acestea sunt depistabile, utilizatorul va fi avertizat.

Erorile care pot apărea în programele STL sunt de două tipuri:

- ➢ erori de concepție;
- ➢ erori de sintaxă;

Erorile de concepție nu pot fi depistate înaintea sau în timpul compilării. Ele țin mai mult de structura programelor şi de modul de concepere al aplicației pe care acestea trebuie să o rezolve. Printre erorile de concepție se numără:

- *bucle infinite*; programul repetă la infinit doar câteva instrucțiuni şi nu are nici un criteriu de ieşire din acel ciclu;
- *existența unei condiții care nu va putea fi niciodată îndeplinită*; programul se va bloca aşteptând ceva imposibil sau nu va executa niciodată un set de instrucțiuni;
- *împărțire la zero* a unui operand în cazul utilizării operațiilor aritmetice;
- etc.

Erorile de concepție se înlătură printr-o scriere atentă a programelor şi cu ajutorul depanării on-line care se va detalia în capitolul 7.

Erorile de sintaxă sunt depistabile la compilare şi se referă la corectitudinea scrierii fiecărei instrucțiuni în parte. Ca exemple de erori de sintaxă se pot da:

- *scrierea incorectă* a cuvintelor cheie;
- *încercarea de a activa o ieşire*;
- *depăşirea limitelor permise ale unor valori numerice* (numărul de temporizatoare este de maxim 255. Încercarea de a folosi temporizatorul T450 va fi semnalată ca eroare).

În mediul de programare FST FEC programele STL se pot verifica din punct de vedere al corectitudinii sintaxei. Verificarea se poate face în *timpul editării* sau *după editare*.

Pentru verificarea sintaxei în orice moment din <u>timpul editării</u>, se apasă tasta F7 (*Extended Commands*) şi se selectează opțiunea *Check syntax*, după care se apasă ENTER, figura 6.1.

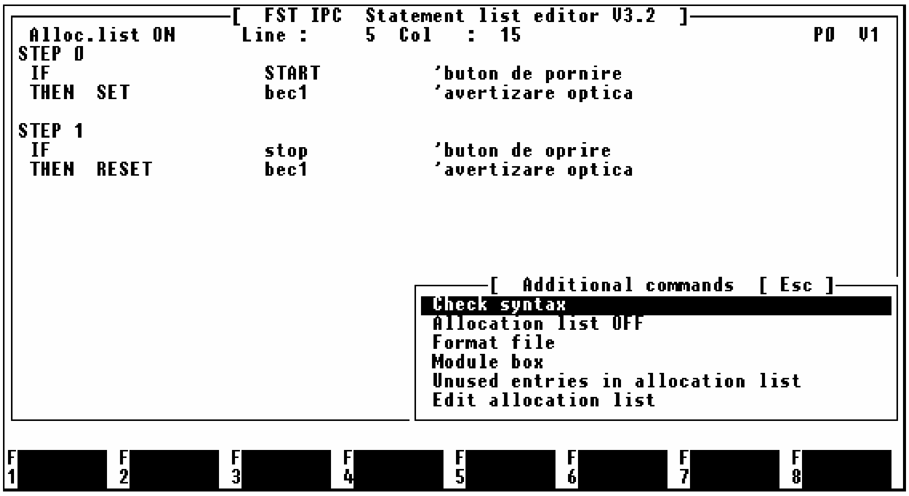


Fig. 6.1. Verificarea sintaxei în timpul editării programelor.

În cazul existenței unei erori (prima întâlnită de la începutul programului), va apare o fereastră de anunțare a tipului erorii iar linia de program în care a apărut eroarea va fi

poziționată în partea de sus a ecranului şi scoasă în evidență cu o dungă în contrast. În figura 2 se poate observa mesajul de eroare apărut la descoperirea liniei de program în care se încearcă activarea ieşirii I0.0 (de la intrări se poate numai citi, nu se poate scrie).

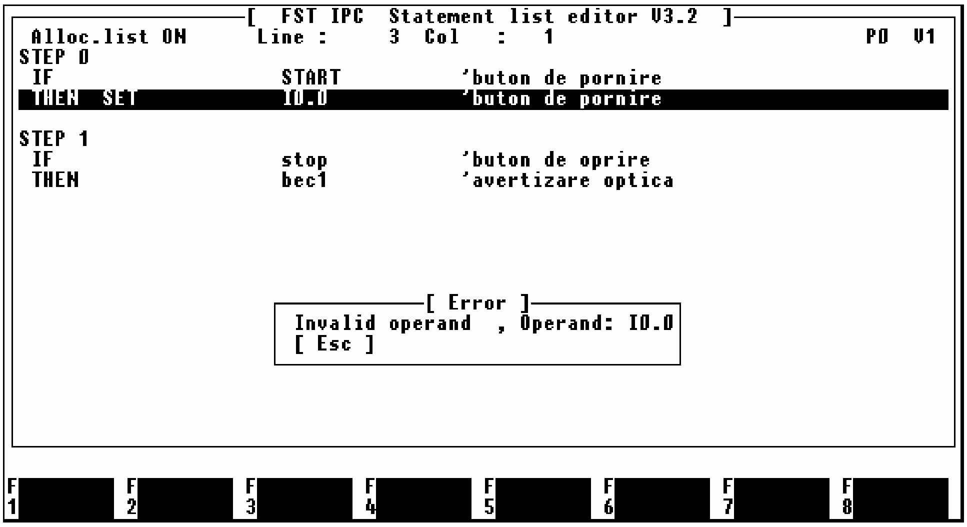


Fig. 2. *Prima eroare descoperită în program. A doua eroare este în pasul STEP 1 unde după THEN şi înainte de "bec1" ar trebui să urmaze SET sau RESET.*

<u>După editare</u> verificarea sintaxei se face cu opțiunea *Syntax test* din submeniul *Statement list*. Va apărea o fereastră cu programele conținute în proiect, fig. 3. Cu ajutorul tastelor direcționale se selectează programul care se doreşte a fi verificat (în cazul nostru P0 vrsiunea 1) şi se apasă ENTER.
Dacă nu există nici o eroare atunci se revine în meniul principal. Dacă există erori va apare o nouă fereastră în care este afişat numărul de erori din program. În exemplul nostru apar două erori aşa cum se vede în figura 6.4.

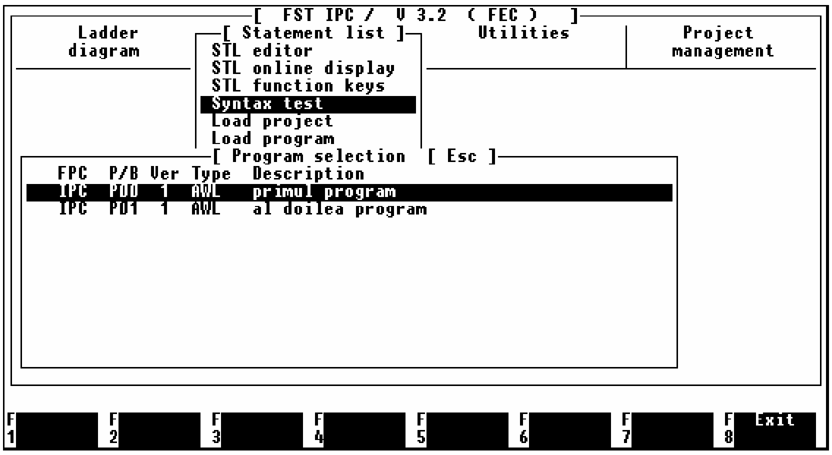


Fig. 3. *Selectarea unui program pentru verificarea erorilor de sintaxă.*

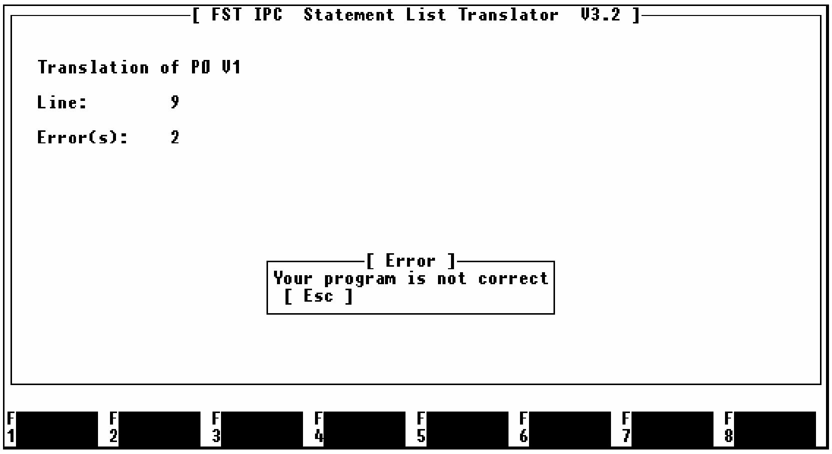
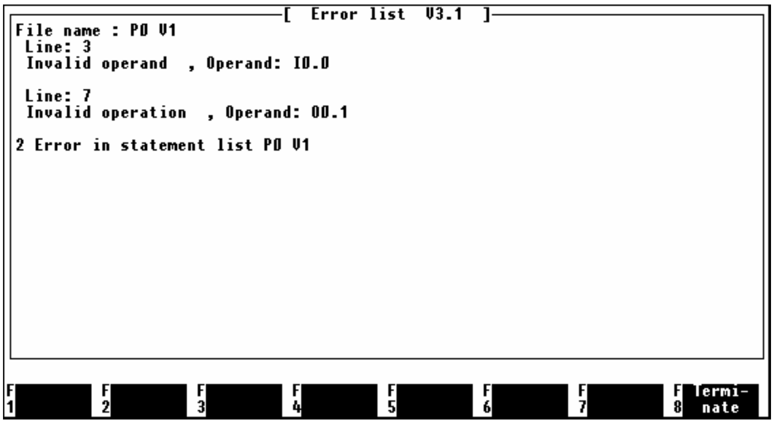


Fig. 6.4. Fereastra de afişare a erorilor din programul P0 versiunea 1.

Pentru căutarea mai uşoară a liniilor din program unde există erori se poate consulta *lista de erori* prin activarea opțiunii *Error List* din submeniul *Statement list*. Apare fereastra de selecție a programului. Figura 6.5. prezintă lista de erori a programului P0 versiunea 1 după selecția acestuia cu ajutorul tastelor direcționale şi apăsarea tastei ENTER.
În lista de erori se dă numărul liniei la care a apărut eroarea şi o descriere a tipului erorii. Aceste informații sunt suficiente pentru remedierea erorilor deoarece, la editarea programelor, în linia de stare din capul ferestrei, se afişează în permanență numărul liniei



la care se găseşte cursorul de editare.

Fig. 6.5. Lista de erori din programul P0 versiunea 1.

Încărcarea proiectului în automatul programabil se execută selectând opțiunea *Load project* din submeniul *Statement list*. Înainte de încărcarea proiectului în FEC trebuiesc îndeplinite condițiile:

- FEC-ul alimentat cu tensiune şi cablul de conexiune cu calculatorul montat;
- Sistemul de operare al FEC-ului trebuie să fie pornit;
- Proiectul trebuie să aibă cel puțin un program;
- În proiect trebuie să existe o configurație a intrărilor şi ieşirilor.

La activarea opțiunii *Load project* va apare o fereastră cu programele, funcțiile modul (CFM) şi modulele program (CMP) conținute în proiect, fig. 6.6. Programele şi modulele care se doresc a fi încărcate în FEC se selectează cu tastele direcționale şi tasta ENTER. Trebuie apăsată tasta ENTER pentru fiecare program care se doreşte a fi încărcat în FEC. Selecția se poate face şi cu mouse-ul. La sfârşitul selecției se apasă tasta F1 pentru începerea operației de încărcare.

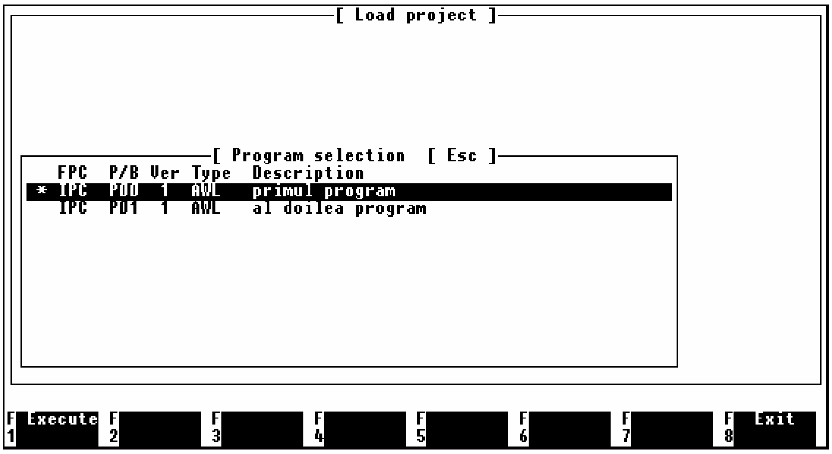


Fig. 6.6. Programele care se pot încărca în FEC. Asteriscul (*) desemnează programele care au fost selectate pentru încărcare în FEC.

La comanda de încărcare a proiectului în automatul programabil se face în mod automat o verificare de sintaxă a fişierelor selectate pentru încărcare. Dacă există programe cu erori, operația de încărcare se anulează şi utilizatorul este anunțat de existența erorilor.
În cazul când nu există erori la această ultimă compilare, se întreabă utilizatorul dacă vrea să încarce în automatul programabil şi fişierele care conțin instrucțiunile STL. Aceste fişiere se numesc *fişiere sursă*. Dacă se selectează opțiunea *N (No)* atunci acestea nu se încarcă, fig. 6.7. Dacă se selectează opțiunea *Y (Yes)* atunci, în viitor, proiectul se va putea încărca din FEC pe oricare calculator care se va conecta la automatul programabil. Operația se poate face cu opțiunea *Upload project* din submeniul *Project management*.

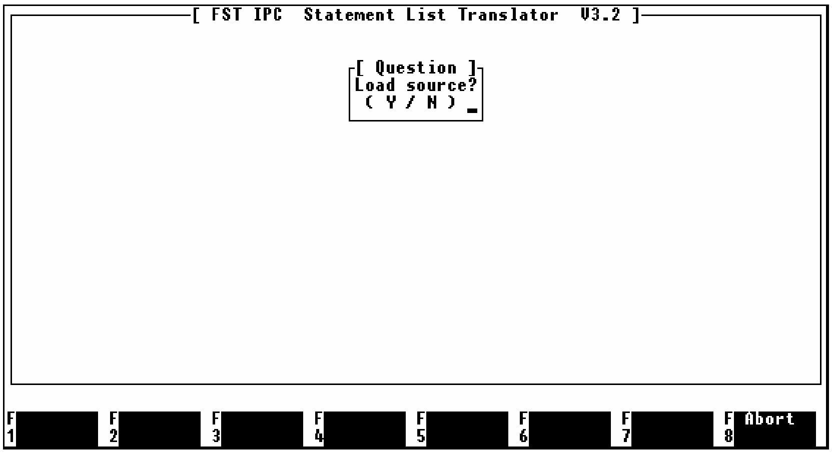


Fig. 6.7. Opțiune de încărcare în FEC a programelor sursă.

După apăsarea tastei F1 fişierele sunt încărcate în FEC unul după altul. Operația de încărcare se poate anula în orice moment prin apăsarea tastei ESC. Dacă se răspunde afirmativ la întrebarea de confirmare a acestei acțiuni, încărcarea se opreşte iar fişierul care tocmai se încărca se şterge din FEC.
Atenție! Varianta veche a acestui fişier NU va fi păstrată în FEC.
Pentru ca proiectul încărcat să devină activ, trebuie ca automatul programabil să fie repornit (cu ajutorul comutatorului de pe partea frontală a carcasei).

UNIVERSITATEA TEHNICA GH. ASACHI IASI
FACULTATEA DE MECANICA
CATEDRA ORGANE DE MASINI SI MECATRONICA
Disciplina: Actionarea sistemelor mecatronice

LUCRARE DE LABORATOR NR. 4

MODULAR PRODUCTION SYSTEM (MPS)– ECHIPAMENT DE INSTRUIRE IN MECATRONICA
Conectarea *on-line* la automatul programabil

1. Scopul lucrarii:

- Crearea abilitatilor de a lucra on –line cu automatul programabil , lucru necesar in supravegherea functionarii automatului, inclusiv in depistarea erorilor de conceptie a programelor

2. Introducere.

Una din caracteristicile unui automat programabil este aceea de a permite conectarea cu o consolă de *programare* şi *diagnosticare*. Consola poate să fie un calculator obişnuit sau unul special dedicat acestei misiuni. De obicei consola este portabilă. Despre programare s-a vorbit în capitolul 6. În acest capitol se vor detalia metodele şi posibilităţile de diagnosticare.

Pentru diagnosticare trebuie să se poată stabili o comunicare între calculator şi AP în timpul <u>funcţionării normale</u> a AP-ului. În acest caz se vorbeşte de *conectare on-line*.
Conectarea on-line este necesară:

1) *Pentru verificarea funcţionării echipamentelor periferice automatului programabil în timpul instalării şi punerii în funcţiune a instalaţiilor de automatizare.*

În acest sens, fiecare senzor pentru semnalele de intrare se poate verifica separat, prin aducerea instalaţiei în starea în care senzorul 'trebuie' să funcţioneze. De asemenea, fiecare ieşire se poate 'forţa' în 1 logic (sau 0 logic) pentru observarea execuţiei corecte a comenzilor date de automatul programabil.

2) *Pentru depistarea erorilor de concepţie din programele complexe.*

Dacă intrările şi ieşirile sunt conectate corect, dar instalaţia funcţionează necorespunzător, atunci există, în programele STL, cel puţin o eroare de concepţie. Această eroare se poate depista prin observarea atentă a stării fiecărui operand în timpul execuţiei programelor.

Acest capitol descrie modul de lucru on-line al mediului de programare FSTFEC cu aplicaţie la automatul programabil E.FEC-20-AC.

3. Depanarea on-line a programelor STL

Urmărirea execuţiei unui program se poate face prin selectarea opţiunii *STL online display* din submeniul *Statement list*. Această comandă va deschide fereastra cu programele proiectului aşteptându-se selecţia unuia din ele. Selecţia se execută cu tastele direcţionale şi prin apăsarea tastei ENTER. În fereastra care urmează este afişat programul STL. Lângă fiecare operand apare valoarea acestuia care este actualizată în permanenţă, pe măsură ce programul se execută în automatul programabil.
În figura 1 se prezintă fereastra de vizualizare a programului P0 (acelaşi de la sfârşitul capitolului 6) după conectarea on-line. După apăsarea butonului 'start' programul a executat pasul 0 în care ieşirea 'bec1' a fost activată cu comandda 'SET bec1'. Programul s-a oprit la pasul 1 unde aşteaptă să fie îndeplinită condiţia de apăsare a butonului 'stop'

pentru a merge mai departe. Se observă că în partea dreaptă a fiecărui operand se afişează valoarea acestuia funcţie de instrucţiunile care s-au efectuat până în acel moment. Linia contrast arată pasul din program care este în execuţie.

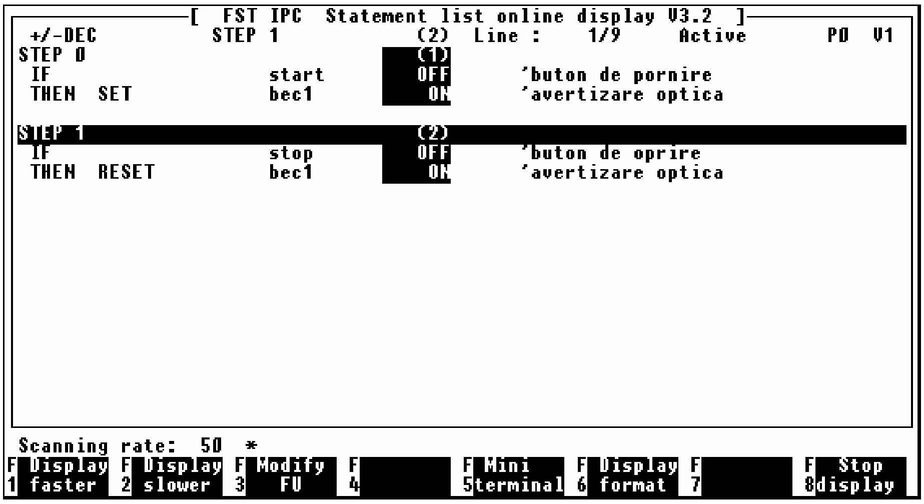


__Fig. 1__. Fereastra de depanare on-line a unui program STL.

Trebuie amintit că informaţiile despre valorile operanzilor şi pasul în execuţie se obţin prin comunicare permanentă cu automatul programabil. Viteza de actualizare a valorilor operanzilor depinde de mărimea programului, adică de cantitatea de informaţii necesară.
Opţiunea *STL online display* oferă posibilitatea de a modifica forţat valorile operanzilor cu ajutorul tastei F3, *Modify FU*. Alegerea operandului care trebuie modificat se poate face cu mouse-ul, cu tastele direcţionale şi apăsarea tastei ENTER sau prin scrierea explicită, prin selectarea tastei F2, *Altern operand*.

3.1. Verificarea on-line a operanzilor de intrare şi ieşire.

Totuşi termenul de *diagnosticare* are un sens mai general decât cel prezentat în paragraful anterior. Astfel, este posibila o conexiune on-line pentru *vizualizarea şi modificarea* operanzilor, chiar dacă utilizatorul nu dispune de programele STL sursă. Opţiunea *IPC online mode*, care se găseşte în submeniurile *Statement list* şi *Utilities*, stabileşte o conexiune on-line 'adevărată' cu automatul programabil. Prima fereastră care apare după selecţie, prezentată în figura 2, oferă posibilitatea de alegere a următoarelor acţiuni:

- tasta F1: vizualizarea şi modificarea operanzilor din AP;
- tasta F3: comunicare cu automatul prin comenzi scrise de utilizator. Modul 'terminal' se utilizează de programatorii avansaţi şi oferă aceleaşi posibilităţi ca şi selecţia F1;
- tasta F4: cu acestă selecţie se şterg toate programele şi toţi operanzii din automatul programabil;
- tasta F7: vizualizează programele şi modulele existente în automatul programabil.
- tasta F8: termină sesiunea de conectare on-line cu automatul programabil.

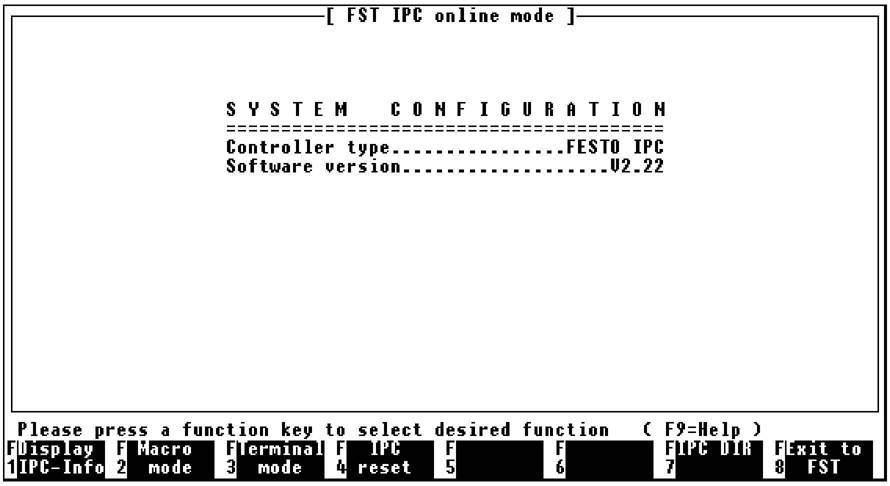


Fig. 2. *Prima fereastră după conectarea on-line cu automatul programabil.*

La apăsarea tastei F1 (Display IPC-Info) apare fereastra prezentată în figura 3. Din această fereastră se pot vizualiza şi modifica: F1 – intrările şi ieşirile; F2 – memoria; F3 – temporizatoarele; F4 – numărătoarele; F5 – regiştrii; F6 şi F7 se discută în paragraful 7.5. Mergând mai departe cu selecția tastei F1 se obține fereastra din fig 7.7. Tastele F1 şi F2 permit vizualizarea intrărilor respectiv ieşirilor *locale* automatului programabil. Cele de *fieldbus* sunt active numai când automatul lucrează în rețea cu alte echipamente 'inteligente'.

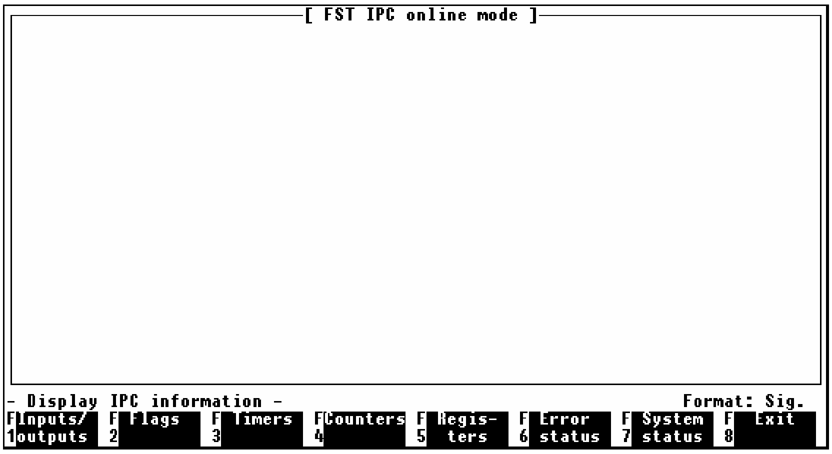


Fig. 3. *Prima fereastra Display IPC-Info care apare la selectarea tastei F1 (vezi şi fig. 2).*

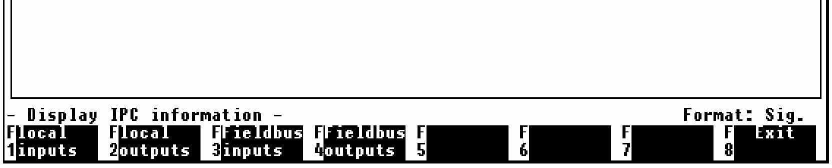
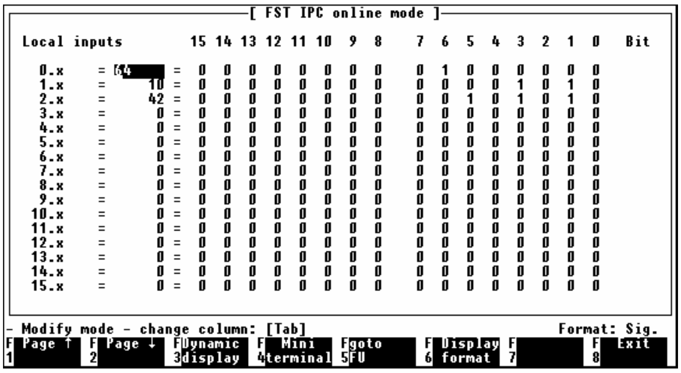


Fig. 4. *A doua fereastra Display IPC-Info care apare la selectarea tastei F1 (vezi şi fig. 3).*

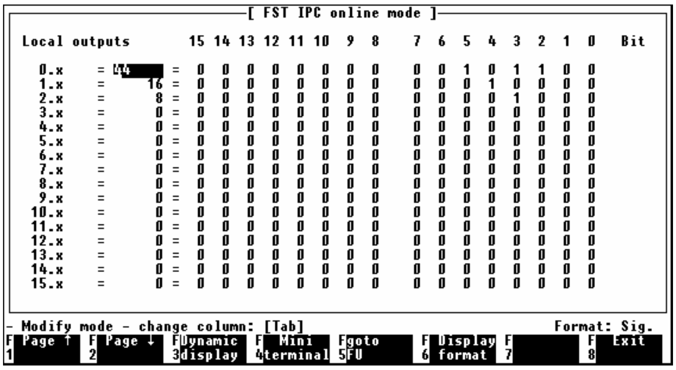
Pentru vizualizarea intrărilor se apasă F1. Pentru vizualizarea şi modificarea ieşirilor se apasă F2. Ferestrele prezentate în fig. 5.a) şi b) vizualizează toate intrările (*inputs*) respectiv ieşirile (*outputs*). Operanzii sunt prezentați ca:

- valoare a cuvintelor (operanzi pe 16 biți);
- valoare a fiecărui bit în parte.

În fereastra intrărilor, fig. 5 a), valorile cuvintelor nu se pot modifica deoarece de la intrări se o poate doar citi. În fereastra ieşirilor valorile cuvintelor se pot schimba de utilizator prin scrierea unui număr în câmpul activ şi apăsarea tastei ENTER. Pentru a ajunge la câmpul dorit se folosesc tastele direcționale.



a) intrări

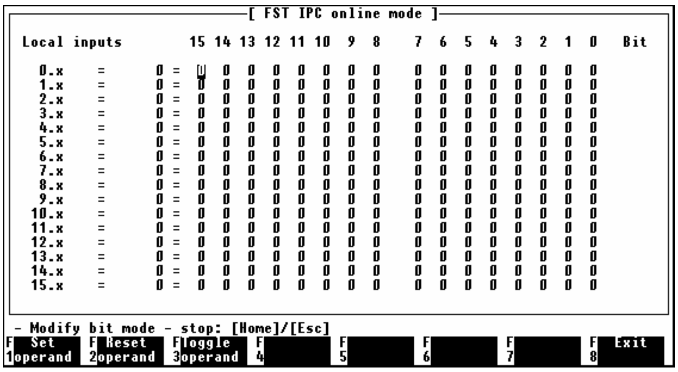


b) ieşiri

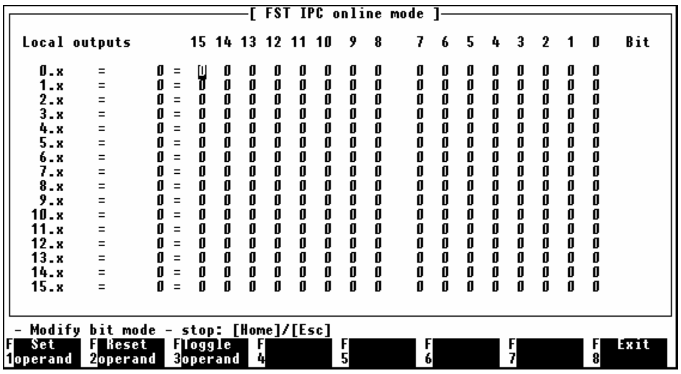
Fig. 5. *Ferestre de vizualizare. Pentru cuvintele de ieşire este posibilă modificarea.*

Din ferestrele prezentate în figurile 5. a) şi b), pentru a schimba valoarea fiecărui bit în parte trebuie apăsată tasta TAB. Meniul de la baza ferestrei se va schimba, fig. 6, iar cursorul de editare se va poziționa în coloanele de biți. În coloanele de biți a ieşirilor se poate modifica orice bit astfel:

- tasta F1 (*Set operrand*): bitul este activat şi devine 1 logic;
- tasta F2 (*Reset operrand*): bitul este dezactivat şi devine 0 logic;
- tasta F3 (*Toggle operrand*): bitul este inversat la fiecare apăsare, adică dacă a fost 1 logic va deveni 0 logic şi invers.



a) intrări



b) ieşiri

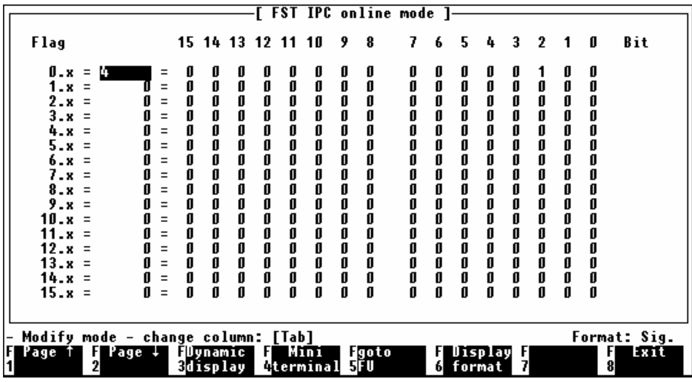
Fig. 6. *Accesul la câmpurile de biți. Biții de ieşire se pot modifica cu tastele F1, F2 şi F3.*

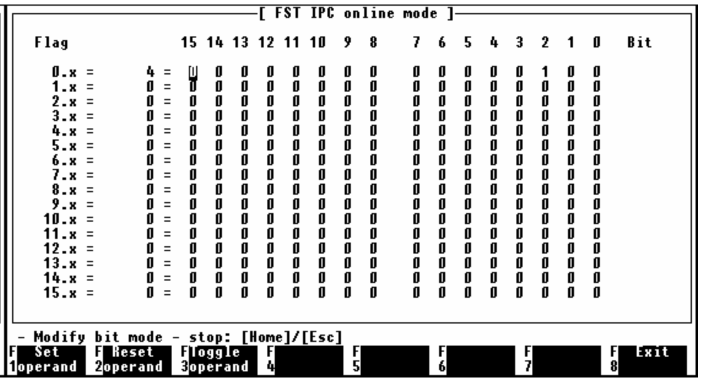
Ieşirea din câmpurile de biți se realizează prin apăsarea tastei ESC. Se va reveni la ferestrele din figura 5. În general terminarea unei acțiuni (închiderea unei ferestre) se realizează prin apăsarea testei F8.

7.4. Verificarea on-line a memoriei, temporizatoarelor şi numărătoarelor.

La fel ca operanzii intrărilor şi ieşirilor se pot vizualiza şi ceilalţi operanzi STL. Mai mult, aceştia se pot modifica foarte uşor de către utilizator.

În figura 7. a) este prezentată fereastra de vizualizare şi modificare a cuvintelor de **memorie**. Se ajunge aici prin apăsarea tastelor F1 şi F2 imediat după conectarea on-line Prin apăsarea tastei TAB se pot modifica biţii memoriei, fig. 7.b).





a) modificare cuvinte memorie b) modificare biţi memorie

Fig. 7*. Accesul on-line la cuvintele şi biţii de memorie.*

În figura 8. este prezentată fereastra de vizualizare şi modificare a celor trei operanzi ce costituie un **temporizator**. Se ajunge aici prin apăsarea tastelor F1 şi F3 imediat după conectarea on-line (vezi şi fig. 3). Se observă că timerul T0 este inactiv (0 logic), că valoarea setată este de 1 secundă (TP0 = 100) şi că timpul a expirat (TW0 = 0). De asemenea se observă că temporizatorul T3 este setat la 20 de secunde (TP3 = 2000). În această fereastră se pot modifica:

- starea timerului, Tn;
- timpul setat, TPn.

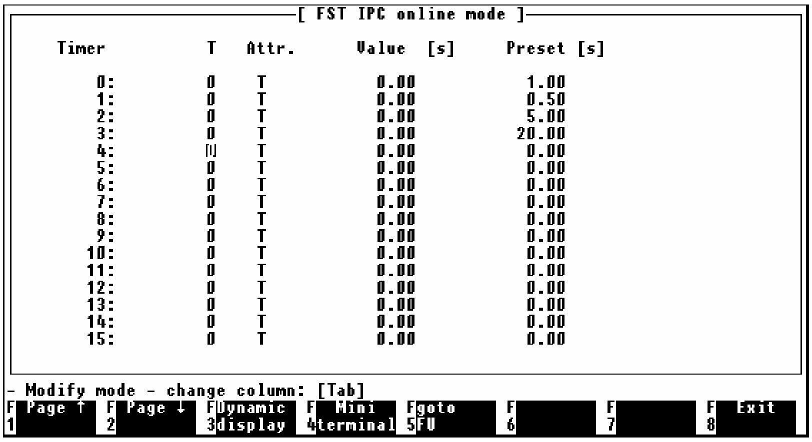


Fig. 8*. Accesul on-line la temporizatoare.*

Cursorul de editare se deplasează cu ajutorul tastelor direcţionale pe fiecare coloană iar între coloane de deplasează apăsând tasta TAB. Stările temporizatoarelor sunt cele existente în automatul programabil în momentul deschiderii ferestrei. Pentru o actualizare permanentă trebuie selectată opţiunea *Dynamic display* (tasta F3).

În figura 9 este prezentată fereastra de vizualizare şi modificare a **numărătoarelor**. Se ajunge aici prin apăsarea tastelor F1 şi F4 imediat după conectarea on-line (vezi şi fig. 3). Deplasarea între câmpurile de date se face la fel ca pentru temporizatoare. Pentru o actualizare permanentă trebuie selectată opţiunea *Dynamic display* (tasta F3).

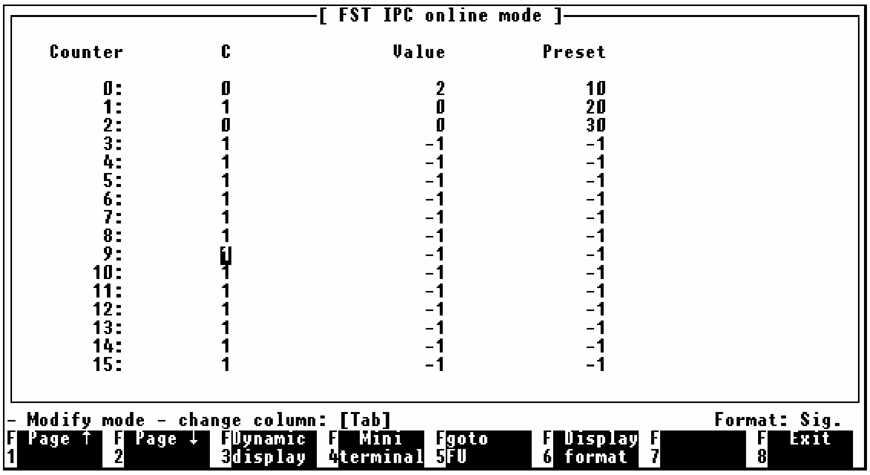


Fig. 9*. Accesul on-line la numărătoare.*

Se pot vizualiza şi modifica toţi cei 3 operanzi ai fiecărui temporizator. De exemplu, în figura 9 se observă că utilizatorul a modificat operanzii primelor 3 temporizatoare, astfel:

- T0 are valoarea preselect, TP0 = 10, a numărat de 2 ori după care a fost dezactivat, probabil cu o comandă 'RESET T0';
- T1 este activ dar nu a numărat nimic din 20, cât are stabilit în TP1.
- T2 este dezactivat şi aşteaptă o comandă 'SET T2' pentru a deveni activ şi a număra până la 30 (TP2).

3.2. Verificarea on-line stării programelor încărcate în automatul programabil.

Dacă în automatul programabil există încărcat un proiect şi acesta este în funcţiune, este posibilă o verificare on-line a stării programelor din proiect. Prin selectarea tastelor F1 şi F7 imediat după conectarea on-line, se deschide fereastra prezentată în figura 7.13. Pentru fiecare program se poate vizualiza starea acestuia (dacă este activat sau dezactivat) şi pasul în execţie în momentul apariţiei ferestrei. Pentru o actualizare permanentă trebuie selectată opţiunea *Dynamic display* (tasta F3).

Din fereastra de vizualizarea a stării programelor, cu tasta F7, se poate activa sau dezactiva fiecare program din AP. Folosirea acestei operaţii asupra programului principal, P0, este echivalentă cu acţiunea comutatorului RUN/STOP de pe partea frontală a carcasei FEC-ului. Programul P0 se va activa şi dezactiva indiferent de poziţia comutatorului.

În exemplul prezentat în figura 10, proiectul din automatul programabil conţine un singur program, P0.

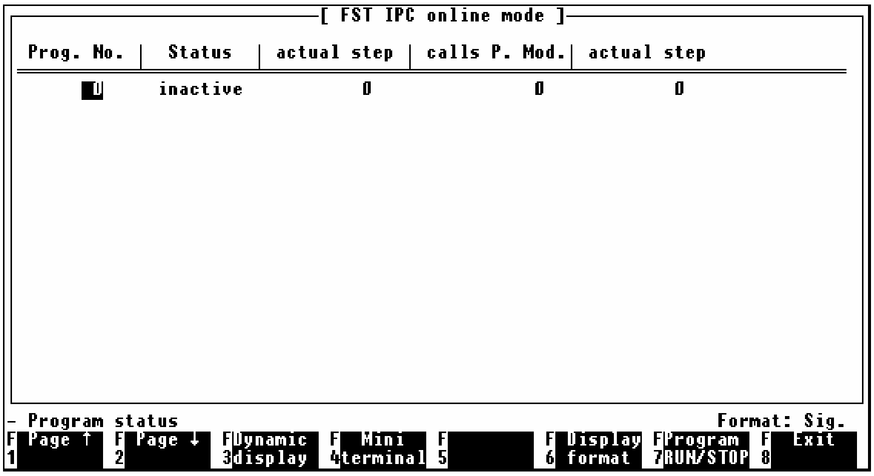


Fig. 10. *Accesul on-line la programele încărcate în AP.*

3.3. Gestiunea fişierelor încărcate în automatul programabil FEC.

Mediul de programare FSTFEC oferă acces la fişierele din automatul programabil.
În FEC există două directoare rădăcină (drive-uri):

- A: - în care se găsesc fişierele cu sistemul de operare.
- B: - ce conține fişierele necesare funcționării proiectului încărcat în FEC.

Accesul la aceste directoare este deplin în sensul că fişierele din ele se pot şterge sau copia din FEC pe calculator sau de pe calculator în FEC.
Vizualizarea fişierelor se face cu comanda *Program Execution* din submeniul *Utilities*. Se mai tastează o dată ENTER după apariția mesajului *File operations*. Va apărea fereastra din figura 11. în care se oferă opțiunile:

- F1 – se vor vizualiza fişierele din automatul programabil, din directorul afişat în partea de sus a ferestrei (în cazul de față A:);
- F2 – se vor vizualiza fişierele din directorul calculatorului afişat în partea de sus a ferestrei.

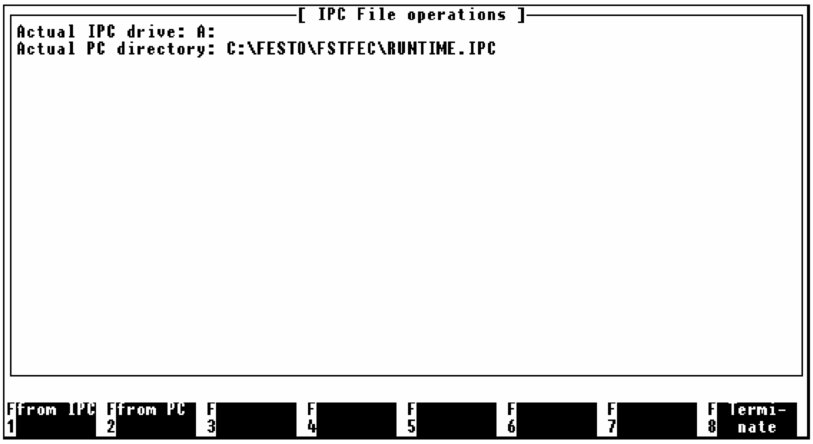


Fig. 11. *Fereastra principală după selecția comenzii Utilities –> Program execution.*

1) În cazul selecției opțiunii *from IPC* (F1) va apărea o nouă fereastră care este prezentată în figura 12.

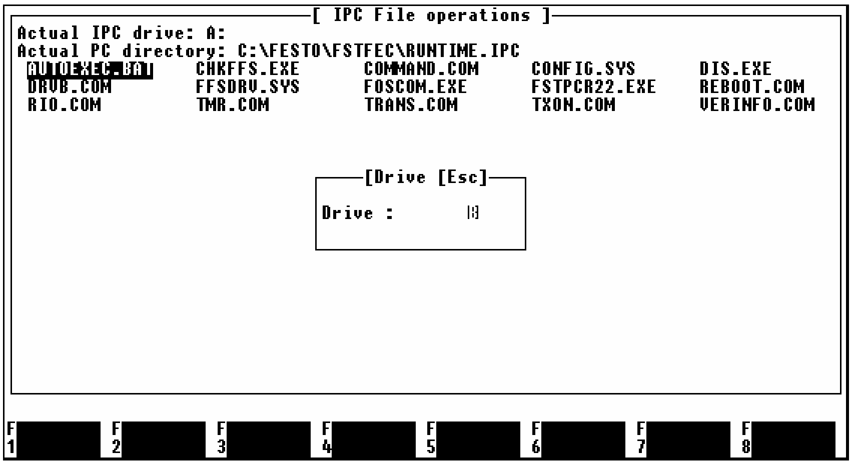


Fig. 12*. Lista cu fişierele conținute în directorul rădăcină A: al automatului programabil*

În această fereastră sunt afişate toate fişierele care se găsesc în directorul rădăcina *A* al FEC-ului. În acest moment sunt posibile acțiunile:

- tasta F1 – şterge fişierul (selectat cu tastele direcționale);
- tasta F2 – copiază din FEC pe calculator (în directorul afişat în partea de sus a ferestrei) fişierul selectat;
- tasta F7 – oferă posibilitatea de a schimba directorul rădăcină al FEC-ului, aşa cum se observă în figura 13.

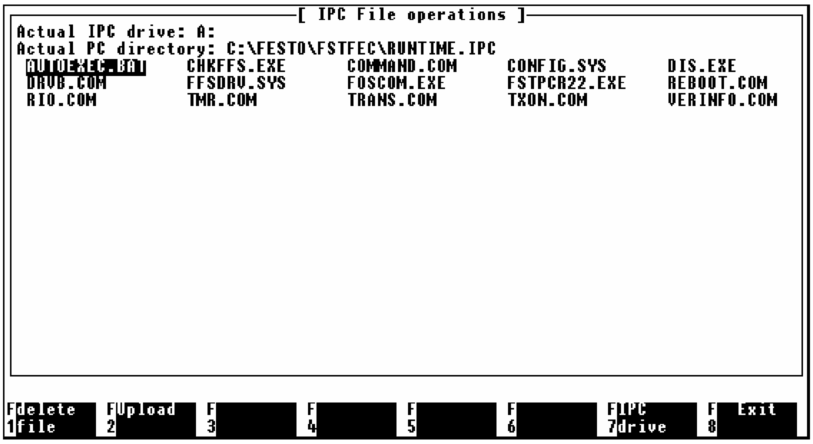


Fig. 13*. Selectarea vizualizării conținutului directorului rădăcină B: al FEC-ului*

2) În cazul selecției opțiunii from PC (tasta F2, vezi şi fig. 11) va apărea o nouă fereastră care este prezentată în figura 14. În această fereastră sunt afişate toate fişierele care se găsesc în directorul calculatorului: "C:\FESTO\FSTFEC\RUNTIME.IPC" (afişat în partea de sus a ferestrei). În acest moment sunt posibile acțiunile:

- tasta F1 (*download file*) – copiază fişierul de pe calculator (selectat cu tastele direcționale) în directorul rădăcină A: sau B: (cel care este scris în partea de sus a ferestrei, după textul: *Actual IPC drive:*);
- tasta F7 (*target drive*) – permite selecția directorului rădăcină A: sau B: unde vor fi transferate fişierele de pe calculator;
- tasta F8 (Exit) – închide fereastra şi se întoarce la cea precedentă.

Trebuie specificat că în fereastra prezentată în fig. 14, se poate selecta orice fişier din orice director al calculatorului (pe drive-ul C:).

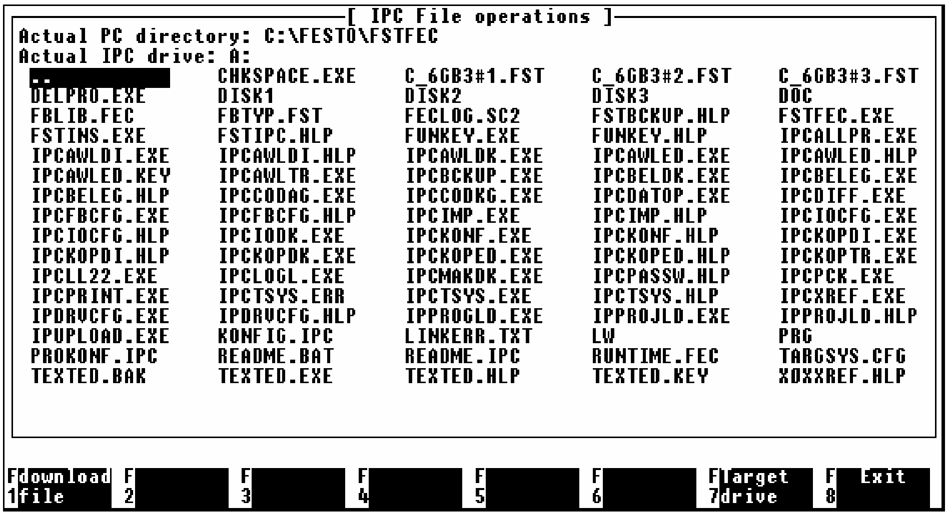


Fig. 14. *Selectarea vizualizării conținutului directorului rădăcină B: al FEC-ului*

3.4. Conectarea în serie a două automate programabile fec

Este posibil ca două automate programabile FEC să funcționeze conectate între ele ca o rețea *master – slave*. În acest caz:

- primul FEC (*master*) are în memorie proiectul care se va executa iar la configurarea intrărilor şi ieşirilor se adaugă opțiunea *Remote FEC* la care se vor seta numere pentru cuvintele de intrare şi de ieşire (distincte de cele folosite pentru intrările şi ieşirile lui);
- al doilea FEC (*slave*) trebuie să fie dezactivat (comutatorul RUN/STOP de pe partea frontală a carcasei este în poziția STOP) şi trebuie şterse din directorul rădăcină B: fişierele "PROJECT.RUN" şi "STARTUP.BAT".

Pentru ştergerea fişierelor din FEC-ul *slave*:

- se montează cablul de conexiune între calculator şi acesta iar comutatorul RUN/STOP se comută pe poziția STOP;
- în mediul de programare FSTFEC se selectează comanda *Program Execution* din submeniul *Utilities*. În continuare se selectează tasta F1 (*from IPC*). Vor apărea fişierele conținute în directorul rădăcină A: al automatului programabil. Pentru schimbarea directorului din A: pe B: se selectează tasta F7 (*IPC drive*) şi se introduce litera 'B' urmată de apăsarea tastei ENTER. După afişarea fişierelor conținute în directorul rădăcină B: se pot selecta fişierele de interes cu tastele direcționale. Fişierul selectat se poate şterge apăsând tasta F1 şi tasta 'y' la întrebarea de confirmare a comenzii.

După ştergerea fişierelor, FEC-ul *slave* trebuie deconectat şi reconectat la tensiunea de alimentare pentru descărcarea din memorie a sistemului de operare.